LONTALK™ Response Time Measurements

March 1992 **LONWORKS™ Engineering Bulletin**

Introduction

This bulletin is intended as a companion document to the LONWORKS engineering bulletins entitled *Enhanced Media Access Control with Echelon's LONTALK Protocol* and *Optimizing LONTALK Response Time.*

The former discusses Echelon's predictive, *p*-persistent CSMA protocol, which provides predictive media access control for collision avoidance, and offers sustained throughput even under conditions of heavy network traffic.

Optimizing LONTALK Response Time describes several of the techniques provided in the LONTALK protocol to improve network response time. These include:

- Designing the network to allow sufficient bandwidth to meet the response time criteria of the application
- Establishing priority for critical packets
- Implementing collision detection hardware for critical nodes
- Sizing the communication buffer pools appropriately
- Choosing the best protocol services by considering both the application and the available network bandwidth
- Setting the retry count, transaction timer, repeat timer, and receive transaction pool size so that no failures are caused by running out of receive transaction records, by incorrect retry counts, or by timer settings that are too short or too long

This engineering bulletin takes a more pragmatic approach to the understanding of response time issues by summarizing the measurements taken in a variety of experiments performed on real networks. The methodology for the experiments is discussed, along with some conclusions that can be reached concerning interactions among the above techniques. These experiments, which constitute only a small sample of those possible, illustrate in particular the effects of using collision detection, priority, and appropriate protocol services to gain improvements in response time.

Interpreting Response Time Data

This bulletin shows application-to-application response time measurements. These data include all of the communication delay, permitting LONTALK application developers to more accurately estimate their system response time. Although application-to-application delays are the only response time numbers that ultimately matter, it is very unusual to find their complete measurements provided for a protocol. Most protocols are incomplete, and require the application developer to augment them with those omitted features necessary for a complete solution to the communication problem. The vendors of such protocols specify response times that often reduce to no more than the packet transit time on the medium. Such benchmarks can appear attractive, but usually prove to be misleading once the system requirements are considered.

This is why comparisons of protocol response time data can be difficult. When evaluating a protocol, care must be taken to add in any communication features that are needed by the application, but not provided by the protocol. Examples to consider include: automatic retries with configurable timers, end-to-end acknowledgements (unicast and multicast), buffer management, duplicate detection, error notification, maintenance of statistics for error detection, independent buffer management and queueing for priority packets, support for multiple channels and large numbers of nodes. Adding such features via the application program can adversely effect response time, often substantially.

Once such services are considered, the true application response time is seen to be much more than a simple measure of transit time on the wire. To use the data in this bulletin in a true comparison with other communication alternatives, therefore, requires a consideration of the complete communication task, rather than casual benchmark numbers for a small portion of the total communication task.

The Response Time Question

Answering the question "What is the response time for this network?" is hard. Unlike the theoretical world of abstract models and Poisson distributions, real networks have far too much individual variation in their designs to permit simple, yet honest answers to broad questions concerning network performance. The following list enumerates only a few network characteristics, changes in which will affect the system's response time (and will also interact with other changes):

- the physical characteristics of the transmission medium and its associated transceivers (distances, transmission rate, noise immunity, collision detection availability, etc.)
- offered traffic loads
- network topology, including the presence of bridges and routers
- the choice of individual message delivery and other protocol services
- the settings of the various protocol service parameters (retry count, repeat timer, transaction pool sizes, data storage buffer sizes, etc.)
- the sizes of groups (as established by node bindings)
- the use of the priority feature
- the response time (resulting from the design) of the application programs

Clearly, then, it is not possible to provide any simple response to the above query.

The experimental measurements listed in this bulletin attempt to characterize two particular classes of network utilization. The first is what might be described as the *best case* situation. This is more easily achieved than others, as the experimenter simply creates the minimum of any deleterious elements (offered traffic load, application program response, etc.), and an abundant supply of the supportive ones (data buffers, channel bandwidth, etc.). The second class of experiments is more characteristic of *worst case,* though clearly not in the extreme (e.g., zero bandwidth). Here *worst case* is interpreted as meaning a situation of extreme network loading (*network saturation,* as defined below).

Actual systems are likely to exhibit performance characteristics falling somewhere between the extremes discussed here. And actual measurements—of the sort illustrated here—are the most reliable way to delineate such characteristics.

Definitions

Protocol Data Units: Modern protocols are hierarchically constructed (the OSI protocol model specifies seven hierarchies, called *layers*), and packets that are transmitted over the network contain information from many of these layers. To simplify discussion of this information, the aggregate data for any particular service are referred to as the protocol data unit (PDU) for that service: thus there are link PDUs, transport PDUs, etc. Of particular interest to the user are the Application PDU (APDU), which is that portion of the packet containing the application data (the *data field*), and the network PDU (NPDU), which is the full packet that is transmitted on the medium (and generally referred to simply as "the packet").

In the NEURON® CHIP implementation of the LONTALK protocol, buffer storage considerations limit the NPDU to 249 bytes. The APDU cannot occupy all of this space, however, as room must be left for address information and additional protocol overhead. In the worst case, assuming largest possible domains and addressing modes, this additional overhead can be as great as 20 bytes (though it is more typically on the order of 8 bytes); thus the APDU is limited to 229 bytes. Of this, one byte is APDU overhead, so—ultimately—the largest data field (contiguous application data, excluding the message type) that can dependably be delivered in a single packet is 228 bytes. This is the most conservative situation; applications will typically not use the worst-case address format (NEURON CHIP serial number), and thus be able to send somewhat larger data fields.

Delay: In general, delay is the interval from the time when an application on one node sends a packet to the time when an application in another node receives the packet. Delay is often further broken down into its individual components of:

- sender application latency,
- outgoing packet formation delay,
- media access control (MAC) delay,
- retransmission delay,
- incoming packet address recognition delay, and
- receiver application latency.

Of these delays, the MAC and retransmission delays both tend to increase with network traffic. As the network traffic increases, packets are delayed in output queues waiting for media access; this delay is known as the MAC delay. Likewise, as the network traffic increases, the probability of not receiving an acknowledgment before timing out (and therefore requiring a retransmission) increases; this additional delay is known as the retransmission delay. The retry packet is typically subject to all the same factors that delayed the initial packet; the expectation in retrying is that the condition that caused the retry was only momentary.

L3 throughput: L3, or Layer 3, is the network layer of the OSI protocol model. L3 throughput is the number of correctly formed packets/second on the physical medium. In many non-LONTALK networks, when the offered traffic exceeds the channel capacity, L3 throughput decreases dramatically due to excessive packet collisions.

L4 throughput: L4, or Layer 4, is the transport layer of the OSI model. L4 throughput is the number of packets/second, excluding retries. In all networks, when the offered traffic exceeds the channel capacity, L4 throughput decreases (due to time-outs and retries). This is often attributable to the L3 degradation, though not always (some non-LONTALK protocols suffer no L3 degradation during overload).

A more general cause of L4 throughput decline is MAC delay. This delay occurs when an application program generates packets faster than the network can transmit them. These application messages then back up in the application and network buffer spaces. Noticing no response from its earlier packets, the application or the protocol software retries. These retries eventually get out, but they take up bandwidth and buffer space from any new packets or acknowledgments. Thus these delays feed upon themselves if the overload condition is not removed. These delays are independent of media access algorithms; time-division multiplexing, token ring, token bus, and all members of the CSMA family exhibit them.

Transaction rate: This is a measure of the maximum number of transactions that can be completed per second. The number of packets in a transaction varies with the transaction type: one packet for unacknowledged, two packets (plus any necessary retries) for acknowledged, and four packets (plus retries) for authenticated protocol services.

Transaction time: This is the inverse of transaction rate, and indicates the time required to complete an individual transaction. Note that this measurement is relative to the source node only, so while an acknowledged transaction time does describe the complete transaction, an unacknowledged transaction time is deemed complete (by the source node) as soon as the message is sent, with no guarantee that it has yet arrived at its intended destination.

Network saturation: This is the point where network delays rise sharply, thus substantially degrading network response time. Different networks behave differently at saturation, but regardless of the network, this situation must be avoided in order to provide good response time to applications. For LONTALK, network saturation occurs at approximately 80% of the transaction rate.

Offered traffic: This metric defines the number of packets per second offered to the network by all the application programs running on all the nodes on a given channel. When the offered traffic exceeds the carrying capacity of the channel, the network saturates and response time degrades dramatically.

Response time: As used in this engineering bulletin, this metric measures the delay between when an application program assigns a value to a network variable or sends a message and when the application(s) in the destination node(s) initiate(s) the control function(s) indicated by the network variable update or by the receipt of the message. This time depends on *all* the above concerns, plus the probability of an error in the receipt of the initial message (which would cause a retry).

LONTALK Response Time Measurement Methodology

Two test beds were constructed to measure LONTALK network response times. The first test bed was designed to measure the best-case throughput of a single node (i.e., assuming that the network is not a bottleneck). The second test bed was designed to contrast the response times of a network operating in both normal and saturated modes.

The Light Loading Measurements (Experiments A, B, C)

For the first class of experiments—throughput of a single node under varying conditions—only two nodes were required, each running a 10 MHz input clock with a network bit rate of 1.25 Mbits/s. The *sender* node sent packets as fast as it could to the *receiver* node. Depending on the protocol service selected, the receiver node either did nothing, acknowledged, or authenticated the transaction. The LONBUILDER™ protocol analyzer was then used to determine the maximum number of packets per second that the sender node could send. From this statistic and from the data field size, the application data transfer rate and the number of transactions per second were computed. Figure 1 illustrates the light loading test setup.

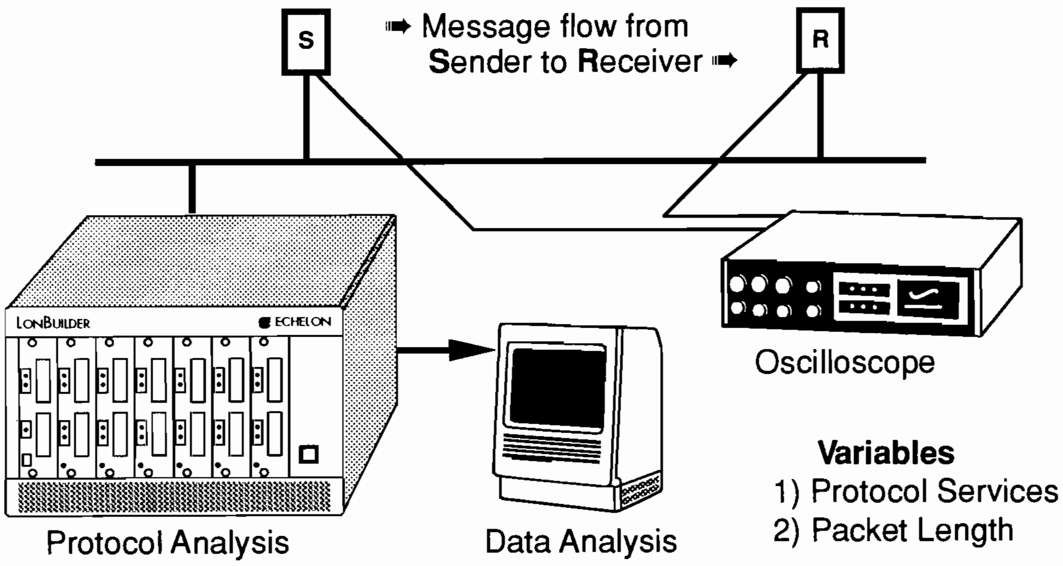

Figure 1. Test set-up for measuring throughput of a single node.

For these experiments, the transaction timer was set to 128 ms. The retry count was set to 2. These timer and retry counts were optimized by means of simple experiments with the test bed. They were each reduced from their defaults until they became too small for the experiment to run without failures. They were then gradually increased back to where no failures occurred. The test command, which sends a network management message to a LONWORKS node, was used along with the protocol analyzer to verify that no failures occurred. These tools, standard within the LONBUILDER development environment, make it easy to verify changes to the protocol timing and retry parameters.

Light Loading Experiments: Results

The light loading experiments were primarily designed to determine the *transaction rates* of a single NEURON CHIP running on a 1.25 Mbits/s LONTALK network. A variety of different data field sizes were used, from one byte (being the minimum value, typical for control messages, and thus typical LONWORKS usage), to 228 bytes (the maximum value, as discussed previously). The total packet size in all cases was 10 bytes greater than the data field size. Transaction rates were measured for each of three different experiments. The net application data transfer rate (in kbits/s) was also computed, in order to illustrate the data transfer capabilities of the LONTALK protocol. As a final step, the sender and receiver programs were modified to toggle the I/O pins of their respective NEURON CHIPs, and true application-to application response times were measured, using an oscilloscope to measure the delay between the I/O events.

Both the transaction and data transfer rates are functions of the bit rate of the channel, the speed of the three processors running on the NEURON CHIP, the size of the data field, and the protocol service used. The three experiments are differentiated by their selected protocol services:

A: network variable updates (results shown in figures 2A, 3A, 4A, 5A) Note that the data fields in these cases never exceed 31 bytes, the maximum size of a network variable.

B: explicit message transactions (results shown in figures 2B, 3B, 4B, 5B)

C: explicit message transactions, no APDU copy (results shown in figures 2C, 3C, 4C, 5C) This is similar to experiment B, but omits the time taken for any assembly of the APDU by the application program; thus it yields results close to those that would be achieved by an external processor using the NEURON CHIP as a communications processor, and communicating with it over a fast parallel port in conjunction with the Microprocessor Interface Program (MIP).

Note that the time taken in the destination node to disassemble the APDU (or the network variable, as appropriate to the particular application) is not accounted for in any of these experiments, as it is not possible to predict the nature of this process (i.e., the measurements only account for the time taken to deliver the messages, and not for their processing by application programs).

For each experiment, results are shown for each of three protocol services: acknowledged, authenticated, and unacknowledged. Note that acknowledged transactions take (at least) twice as many messages to complete as those that are unacknowledged; authenticated transactions take twice as many again.

Also, the time to acknowledge a message increases with the size of the message being acknowledged. This is because an incoming packet is not acknowledged until all needed system resources (to ensure that the packet will be delivered to the application) are allocated. The most time-consuming portion of this operation involves allocating a receive transaction record, allocating an application buffer, and copying the entire incoming packet into this buffer. This is an important aspect of the LONTALK protocol with respect to reliability: it prevents packets from being acknowledged, yet never being delivered to the application due to lack of system resources.

Figure 3A is of particular interest, as it represents one NEURON CHIP sending unacknowledged packets to another NEURON CHIP as fast as it can. In this and similar diagrams, of course, the transaction rate decreases as the number of bytes per packet increases. Note also that the value axes on these diagrams change to suit the exhibited data.

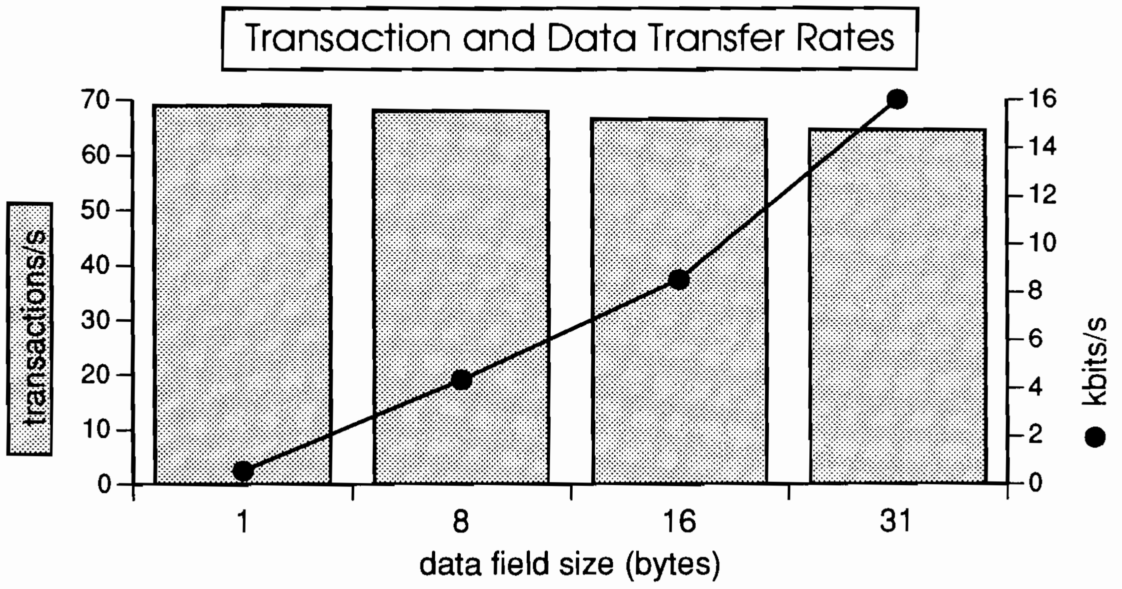

Figure 2A. Network Variable Updates, Acknowledged.

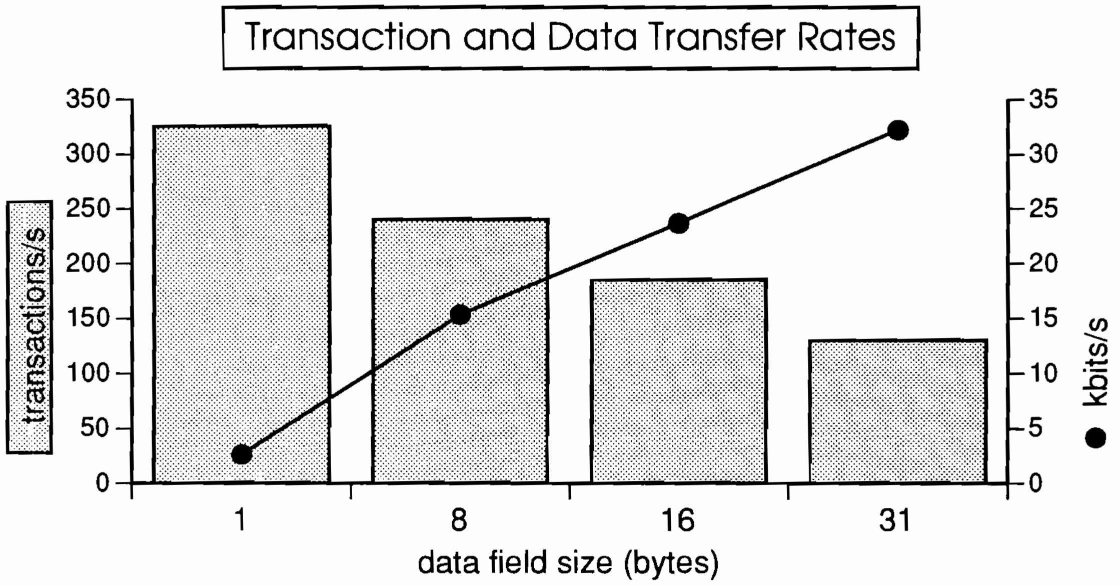

Figure 3A. Network Variable Updates, Unacknowledged.

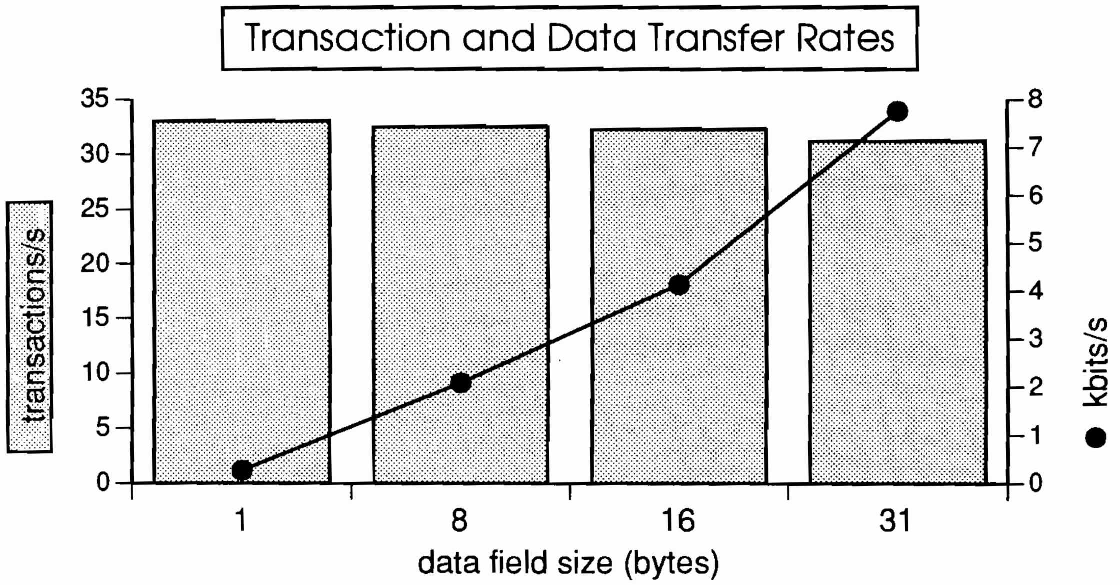

Figure 4A. Network Variable Updates, Authenticated.

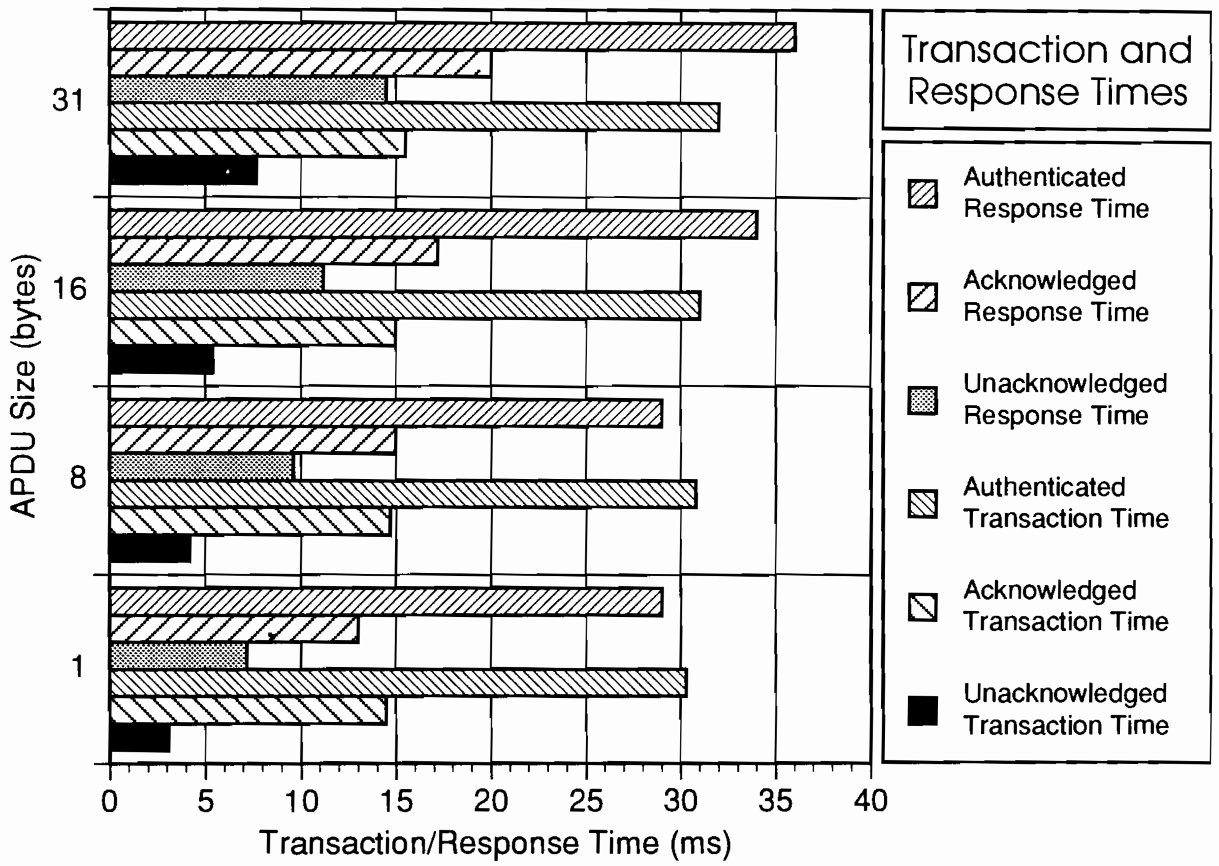

Figure 5A. Network Variable Updates.

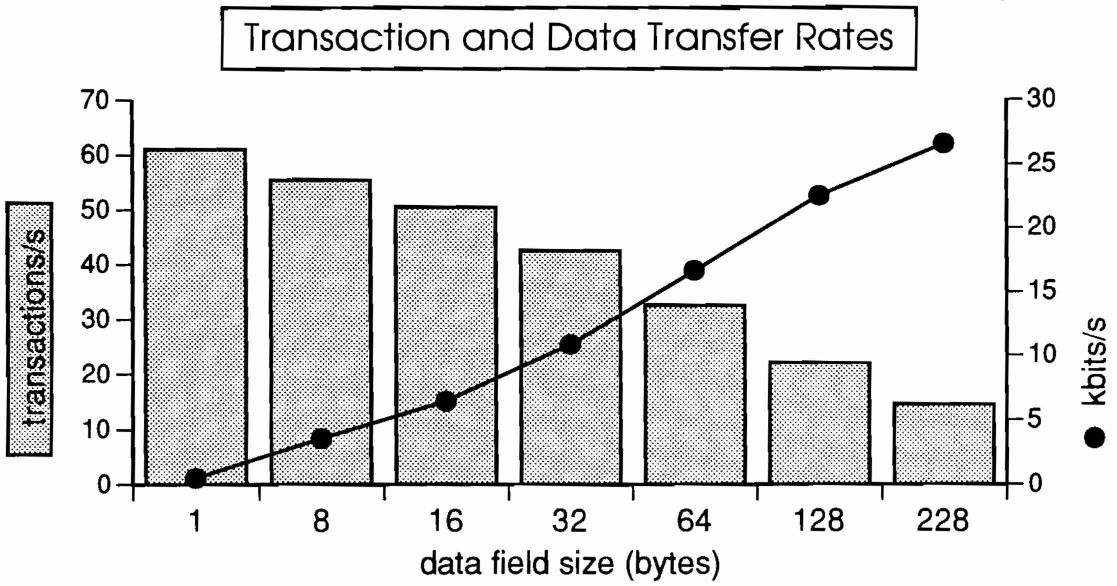

Figure 2B. Explicit Message Transactions, Acknowledged.

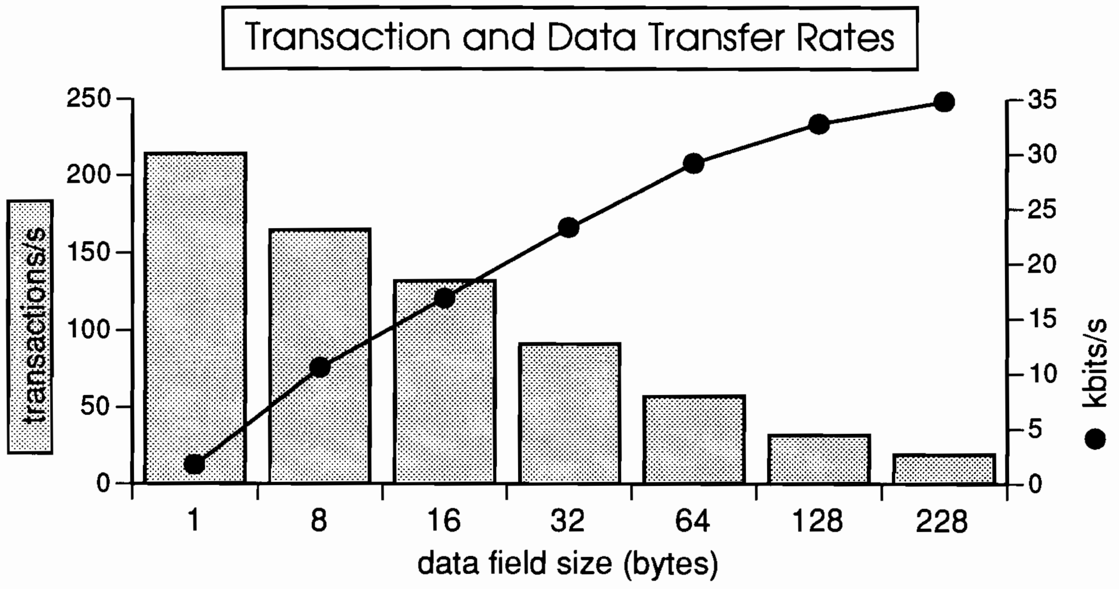

Figure 3B. Explicit Message Transactions, Unacknowledged.

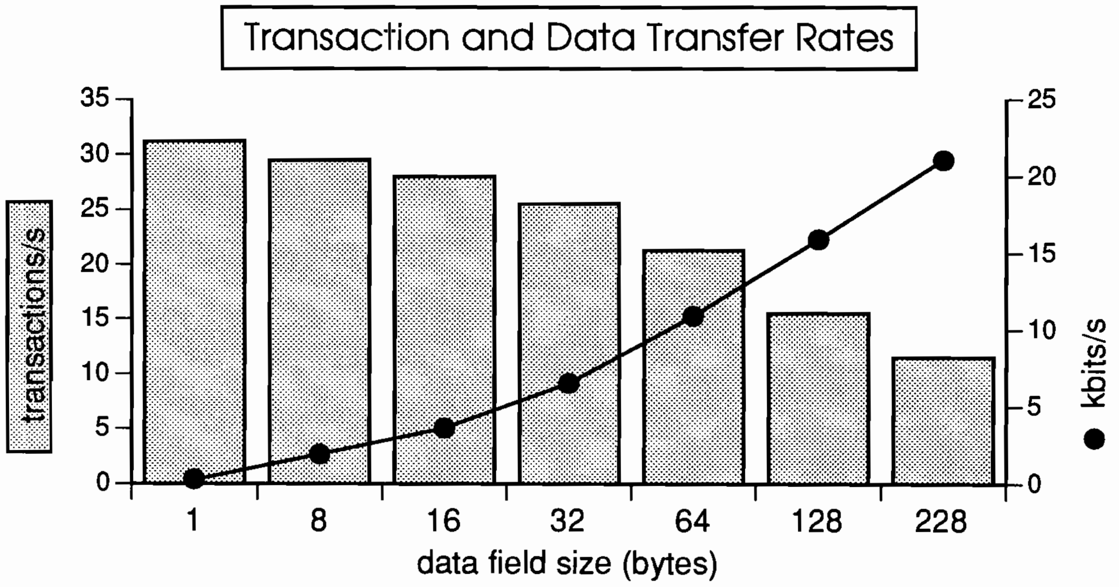

Figure 4B. Explicit Message Transactions, Authenticated.

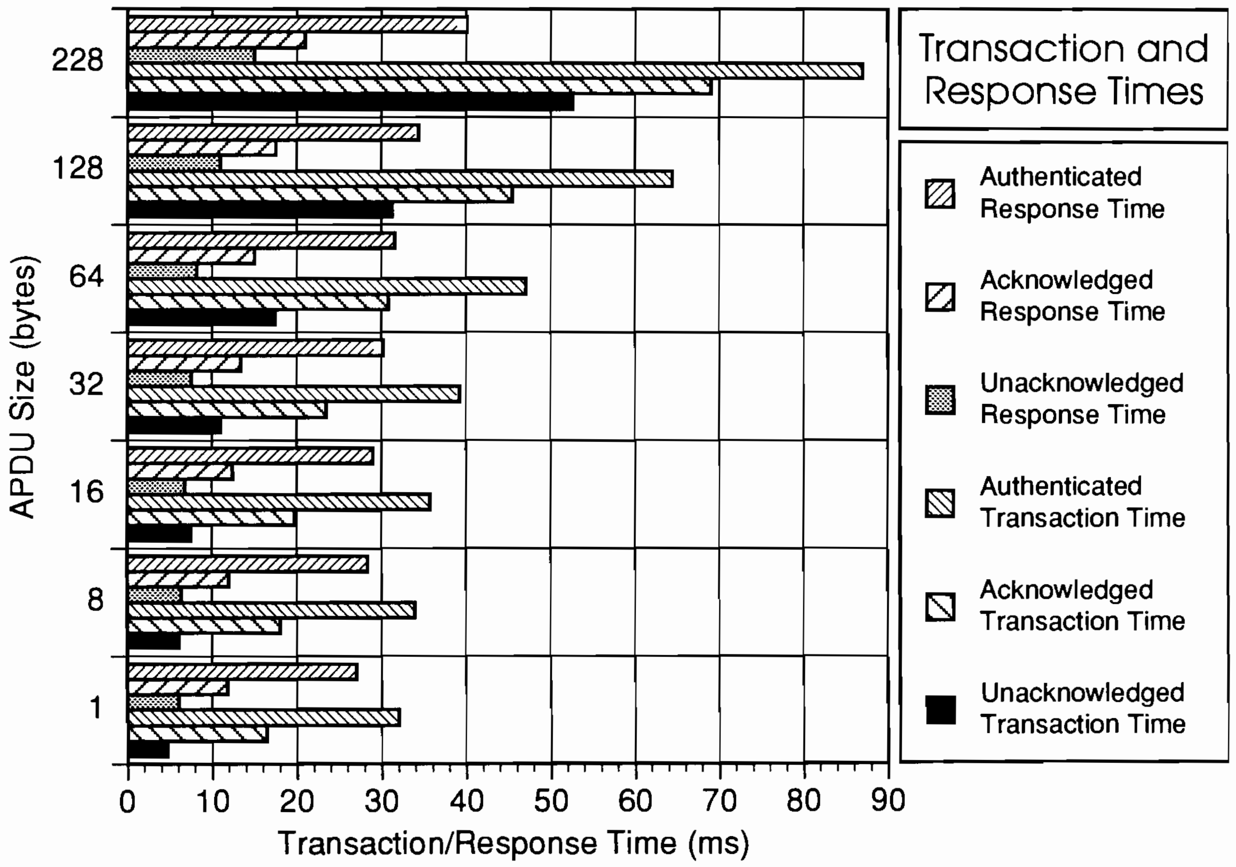

Figure 5B. Explicit Message Transactions.

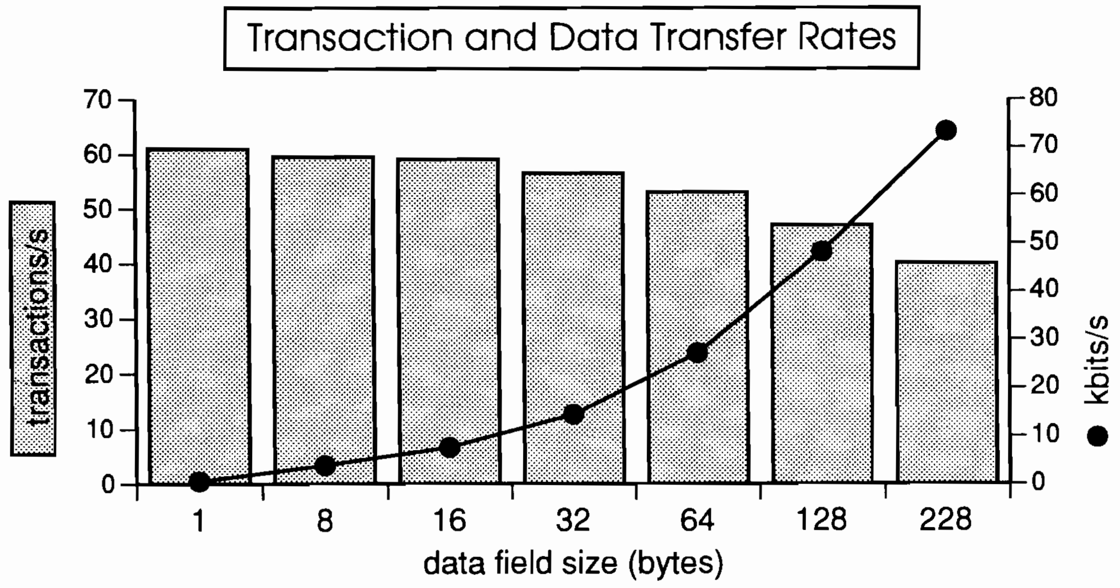

Figure 2C. Explicit Message Transactions (no APDU copy), Acknowledged.

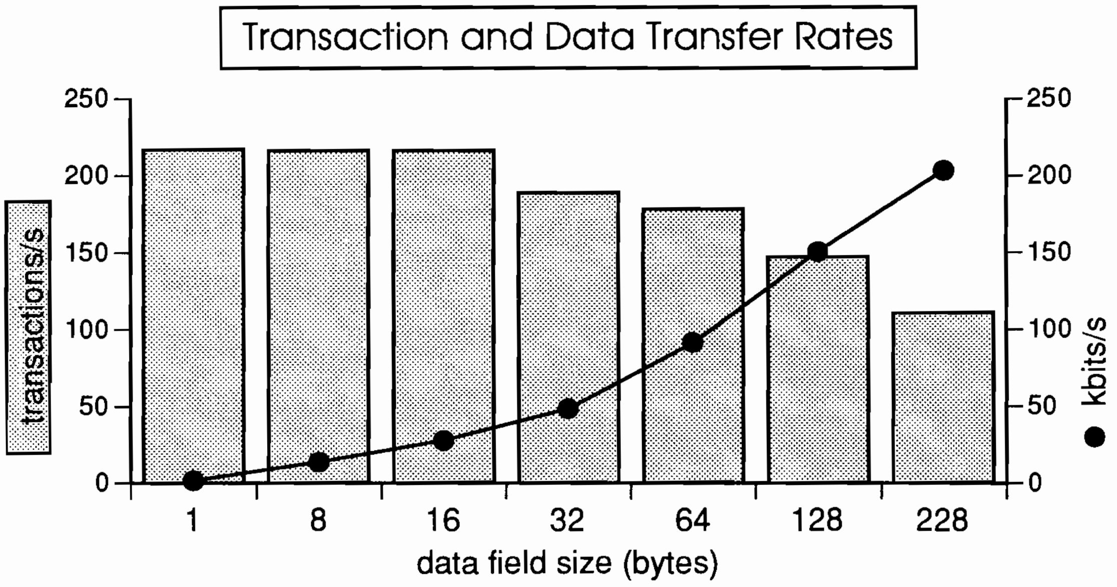

Figure 3C. Explicit Message Transactions (no APDU copy), Unacknowledged.

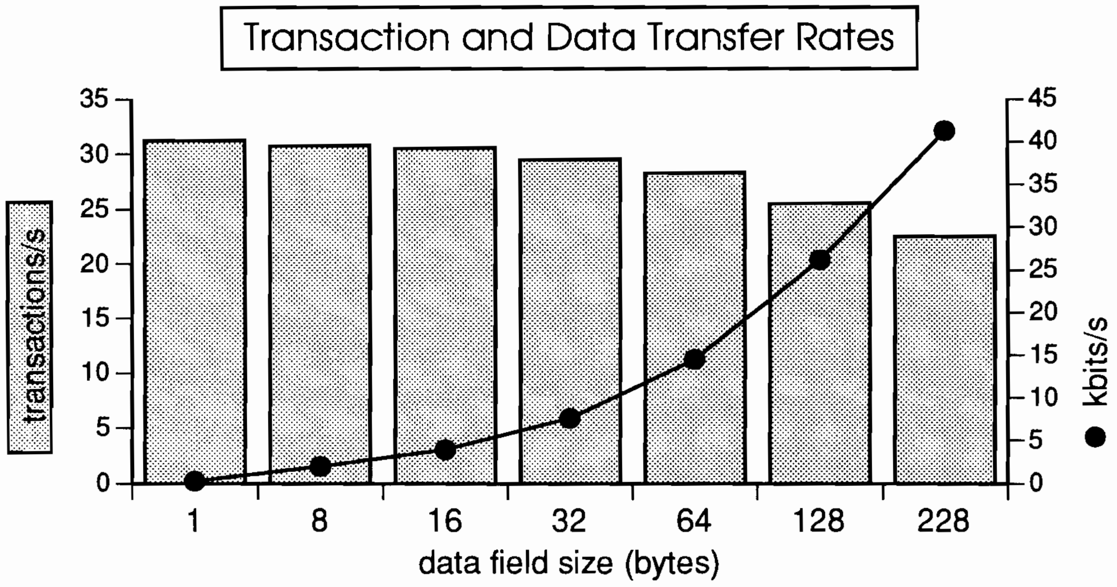

Figure 4C. Explicit Message Transactions (no APDU copy), Authenticated.

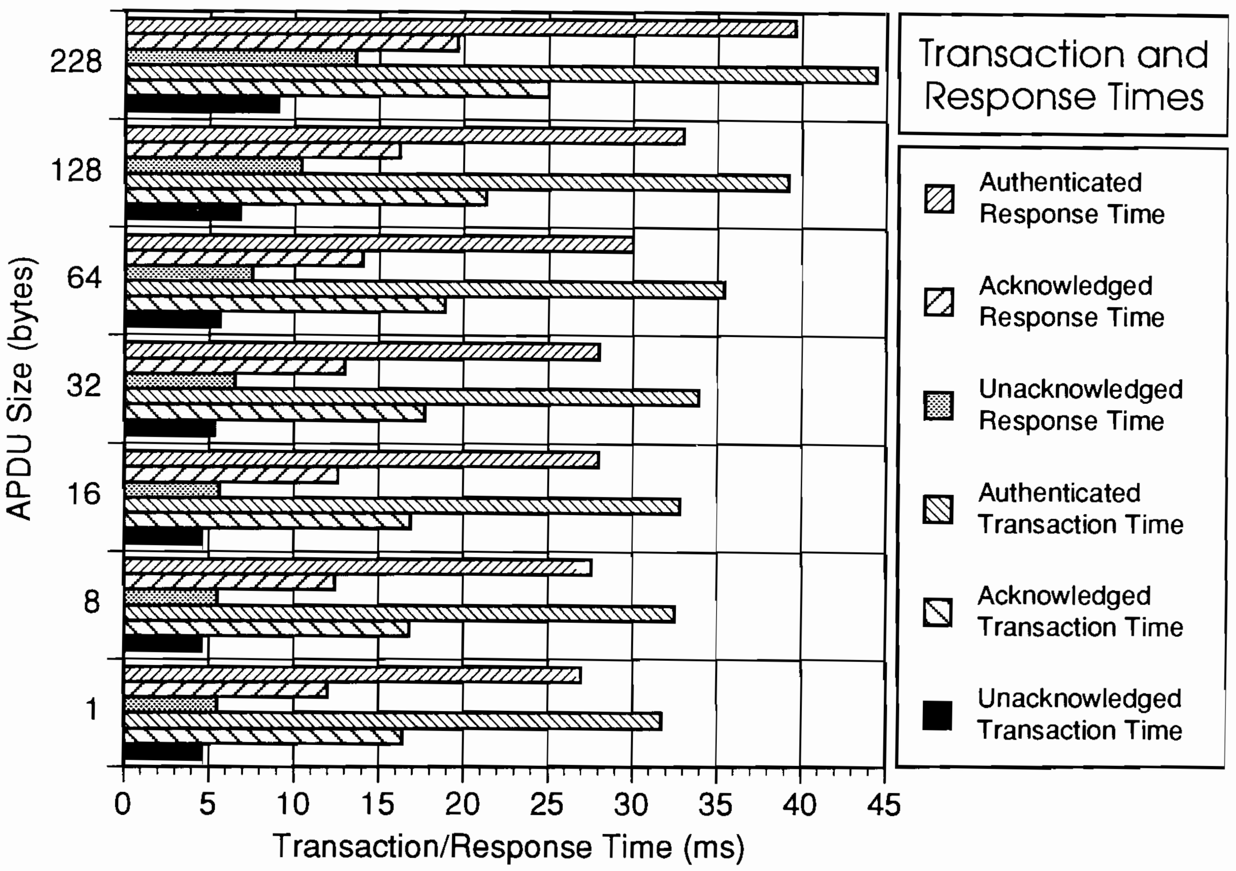

Figure 5C. Explicit Message Transactions (no APDU copy).

Light Loading Experiments: Conclusions

The maximum transaction rate of a single node is 326 transactions/second, using one-byte, unacknowledged network variable updates at 1.25 Mbits/s. The minimum response time between nodes is 5.5 ms, for explicit messages up to about 10 data bytes. The maximum application data transfer rate is 203 kbits/s, using 228-byte, unacknowledged explicit messages in conjunction with an external processor (if the message data are initialized and copied into the application buffer by the NEURON CHIP, this rate drops to 35 kbits/s).

Network variable update response times are somewhat slower (18-90%, increasing with the data field size) than those of corresponding explicit message transactions. This is due to the additional data copy necessary at the receiver end.

The Extreme Loading Measurements (Experiments D through I)

For this class of experiments—network performance under extreme loading conditions—thirty-four nodes plus a network manager and protocol analyzer were employed. Thirty-two of these nodes were loaded with NEURON C programs written to generate programmable levels of background traffic. Two nodes (again designated *sender* and *receiver*) served as both the nodes under test and the accumulators of the statistics. Figure 6 illustrates the extreme loading test set-up.

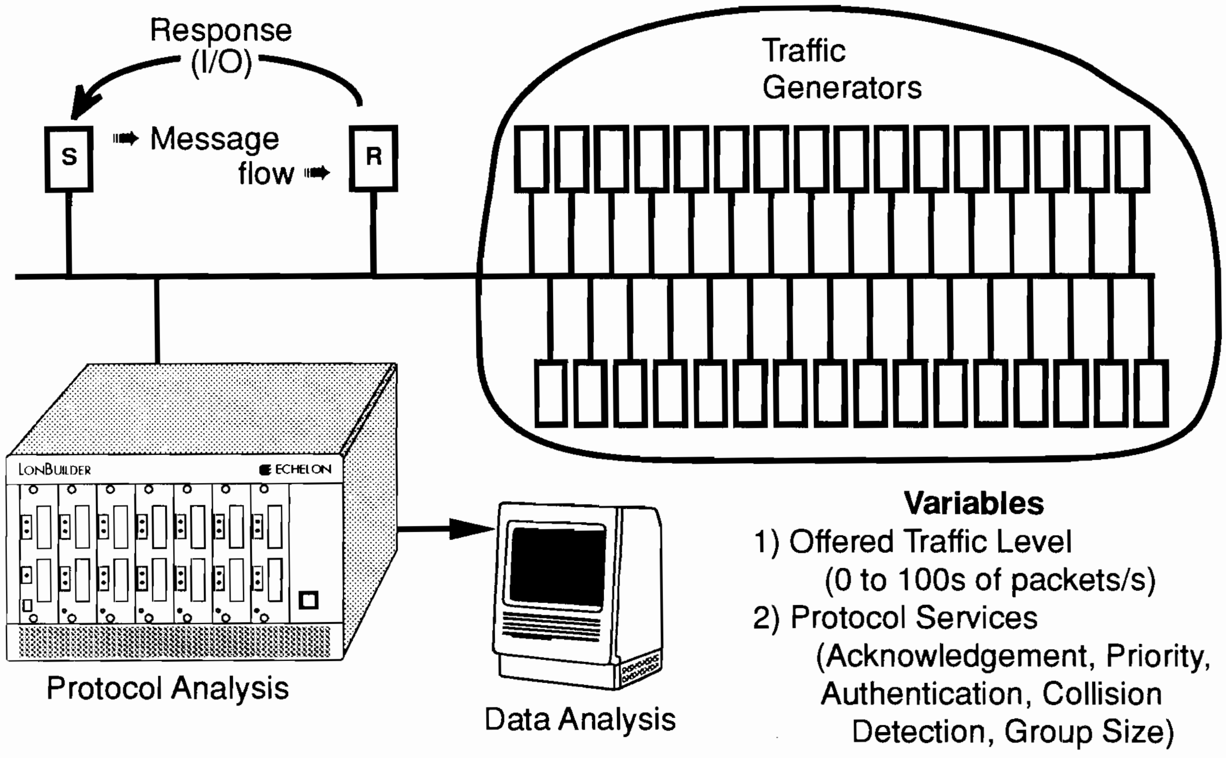

Figure 6. Test set-up for measuring throughput under extreme loading.

For these experiments, a special program (running on the LONBUILDER PC) commanded the sender node to initiate testing at a particular traffic level for a duration of ten minutes. The receiver node then commanded the other thirty-two nodes to begin generating traffic, such that the aggregate number of packets equalled the desired "offered traffic" load. During each ten-minute testing interval, the receiver node ran 200 separate response time tests.

Each test consisted of the sender node starting a NEURON C timer (see the *NEURON C Programmer's Guide* for information on timer accuracy) and sending a packet to the receiver node. When the receiver received the packet, it changed the state of one of its I/O pins, which in turn was wired to an I/O pin on the sender node. When the sender detected that the I/O pin had changed state, it knew that the receiver node had received the message and acted upon it. The value of the timer at that moment was then the response time for that test (this value was adjusted by the sender to account for the time necessary to detect the asserted I/O pin and stop its timer).

If, during the testing, the protocol reported that a transaction failed, then the timer was stopped and a failure logged. Once the 10-minute testing interval had expired, the sender node commanded the traffic generator nodes to stop. It then individually polled the 32 generators to determine how many packets they actually sent. This was used for the L4 throughput numbers. Finally, the protocol analyzer was polled to get the L3 throughput numbers and the packet error rate. During these tests, all CRC errors were assumed to be collisions.

Extreme Loading Experiments: Results

Network performance was measured under a variety of conditions, adding and/or excluding priority, collision detection, authentication, and collision avoidance as necessary to observe their effects on responsiveness. The experiments corresponding to these conditions are referred to as experiments D through I in the sections on the following pages.

In all cases, the data rate chosen was 78 kbits/s, using a 10 MHz input clock. The packet size was 11 bytes—typical for the updating of a one-byte network variable. As with the previous experiments, the transaction timers, receive timers, retry counts, and numbers of input communication buffers were adjusted to maximize throughput and minimize the number of retry packets (thus improving L4 throughput). With unicast messaging, this can be done by examining the number of acknowledged messages sent (using the protocol analyzer); if this is close to the number of acknowledgments sent, then few retries have occurred. With multicast messaging, separate retry messages are enumerated. Retry and communication buffer count values were verified using the network management status command (invoked via the test button in the LONBUILDER application node target hardware screen) to extract error statistics from the nodes. In all experiments, these were adjusted such that there were no transaction failures and no packets lost due to lack of buffers.

Each response time experiment is accompanied by a brief description of the experiment and most include a set of three charts:

- offered traffic vs. delay
- 200-sample response time histograms
- L3 and L4 throughputs versus collision rate.

For experiment D, for example, these are figures 7D, 8D, and 9D respectively. Experiments H and I show only the collision rate chart, using a different scale.

To increase readability, only two runs appear on the histogram figures. The first run shows response time data when the network is just below saturation, an offered traffic level of 300 packets/s (at a 78 kbits/s data rate). The second shows response at an offered traffic level of 500 packets/s, when the network is well above saturation (for a 78 kbits/s data rate, with an average packet size of 11 bytes, network saturation occurs at 360 packets/s). These levels are of particular interest, as MAC delays play a bigger role when the network is saturated. The histograms illustrate an important characteristic of the LONTALK protocol: the average response time curve is nearly flat from zero offered background traffic up to very near saturation.

Experiment D (Baseline Network Performance)

This experiment established a reference performance level, using default services and unicast messages. With the reference thus established, the later experiments demonstrate the effects on performance of using unacknowledged messages, authentication, priority, collision detection, and multicast groups.

In experiment D, all the background traffic generators generated multicast acknowledged messages, using a group size of two (thus each of the traffic generator nodes sent each message to one other such node). No priority was used. The transaction timer was set to 128 ms. The retry count was set to 4. The receive timer was set to 2048 ms.

As seen from the graph in figure 7D (which shows the offered traffic versus delay for this experiment), the average response time remains linear at about 15 ms. This flat average response time is maintained, increasing only modestly with load until the network approaches saturation, at which time it increases to about 200 ms. Note, incidentally, that each of the best and worst case samples shown in these graphs is typically only one among the 200+ samples at each traffic level, and as such cannot form a basis for any general correlation rule (for example, even though figure 7D shows a higher maximum delay at the 100 packets/s level than at the 200 level, one cannot conclude that increasing the offered load will reduce the maximum delay time!).

Figure 8D shows a histogram plotting the distribution of the delay samples over the range of measured delay values (as discussed previously, only two traffic levels are shown).

At the 300 packets/s network traffic level, the histogram shows two response time clusters. A well-defined cluster occurs around 15 ms; a second less-defined one around 120 ms. The second cluster results from retries, when the initial packet is not acknowledged until after the timer expires.

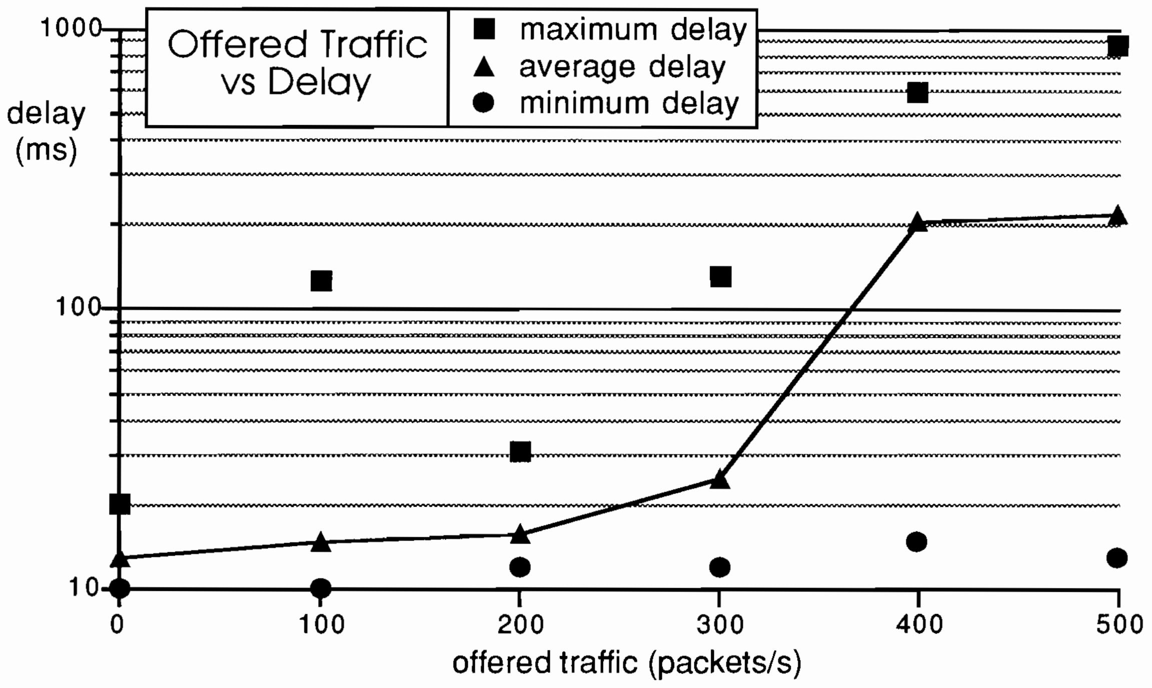

Figure 7D. Acknowledged, no Priority, no Collision Detect.

At the 500 packets/s network traffic level, the clusters are not so well-defined. This is because the network is saturated. As mentioned earlier, MAC delays come into play when the network is saturated, causing a more scattered distribution of response times.

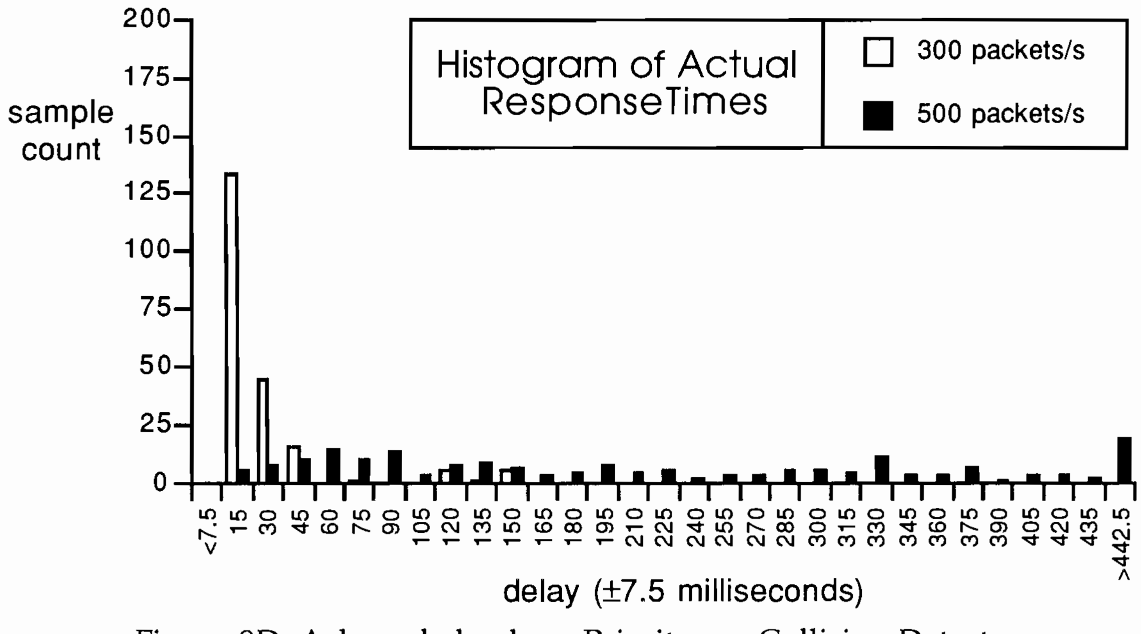

Figure 8D. Acknowledged, no Priority, no Collision Detect.

Figure 9D shows L3 and L4 throughputs versus the collision rate for the different traffic levels. Note that, unlike traditional CSMA approaches, the LONTALK protocol only slightly degrades L3/L4 throughput as the network saturates. This is due to the collision avoidance algorithm in the LONTALK protocol, which controls the collision rate below network saturation, and tends to minimize the collision rate even when the network is driven to saturation for long intervals. This collision avoidance gives the LONTALK protocol its unique combination of throughput similar to that associated with token-passing MAC layers (even under heavy load), and the cost and multimedia benefits of CSMA MAC layers.

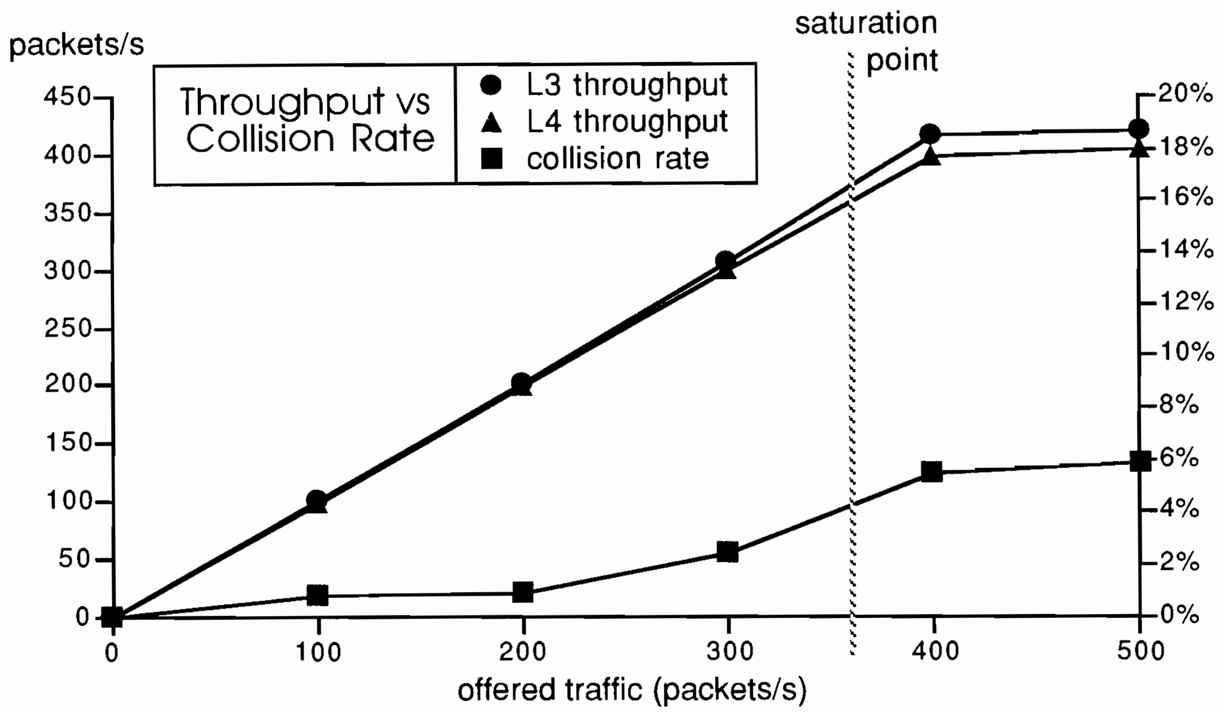

Figure 9D. Acknowledged, no Priority, no Collision Detect.

Experiment E (Effect of Priority on Response Time)

For the priority experiment, the baseline configuration (experiment D) was modified so that the two nodes under test were assigned priority. To do this, two priority slots were configured on the channel. The first priority slot was assigned to the "request" node, the second slot to the "reply" node. In addition, priority message service was specified for the sender's test packet. The characteristics of the traffic generator nodes, and all timers and retry counts remained unmodified. The original tests, re-run under these conditions, yielded the charts shown in figures 7E, 8E, and 9E.

Figure 7E illustrates that the use of priority keeps the average response time at the same level regardless of load. This is in contrast to the previous experiment (figure

7D), where the average response time increased from 15 to 200 milliseconds as the network entered saturation. The effect of priority on response time is even more dramatically illustrated by figure 8E (compared with figure 8D), which shows almost all transactions completing in less than 15 ms.

Figure 7E also shows that while in most cases the worst-case response time was measured around 20 ms, in two instances it was measured at roughly 120 ms (because the initial transmission attempt failed due to a collision). These instances show that the use of priority does not eliminate the possibility of collisions. The two maximum delay samples that occurred (at offered traffic levels of 200 and 300 packets/s, well below the saturation traffic level for the network) resulted from the fact that the nodes did not remain synchronized after the transmission of the previous packet was completed.

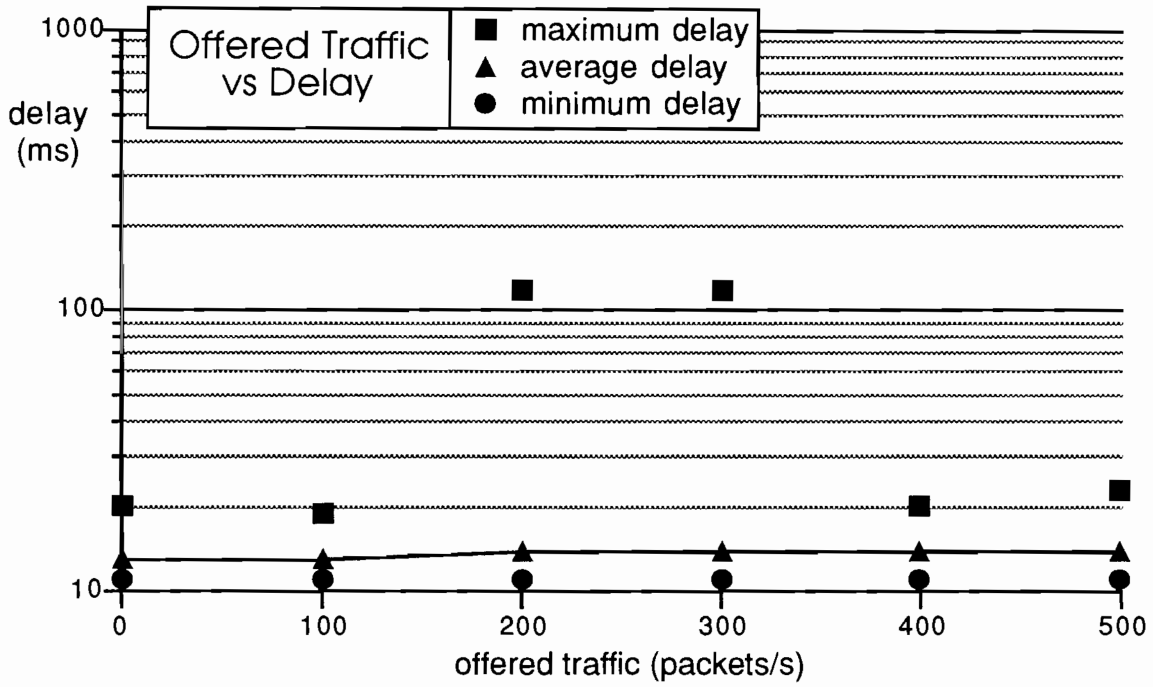

Figure 7E. Acknowledged, Priority, no Collision Detect.

Nodes remain synchronized only for the duration of the priority slots, plus sixteen randomizing slots, after a packet ends. If one of the nodes does not transmit during this period, the network becomes idle, and the concept of a dedicated priority slot no longer has meaning. In such cases, when a priority packet is delivered to the output queue of the NEURON CHIP, the packet is sent out using the *non-priority* MAC algorithm. If another (priority or non-priority) node decides to send a message at the same time, the result is a collision. Having collision detection hardware on the priority nodes avoids the resulting delay, because a node so equipped would detect the collision immediately, and retransmit without waiting for its transaction timer to expire. Even without collision detection, however, the use of priority effectively reduces the maximum response times.

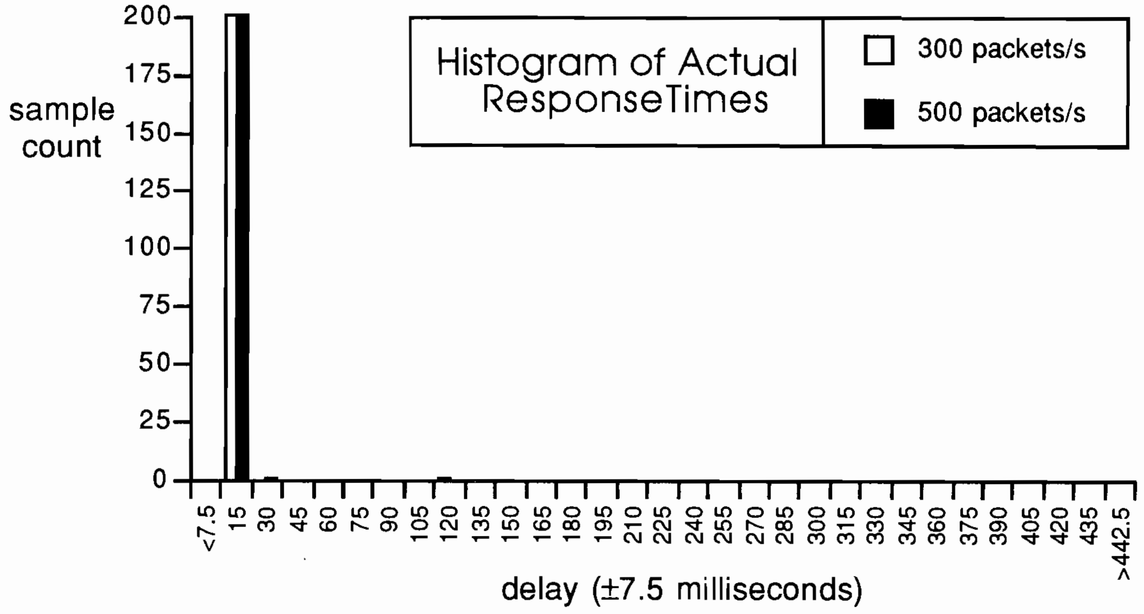

Figure 8E. Acknowledged, Priority, no Collision Detect.

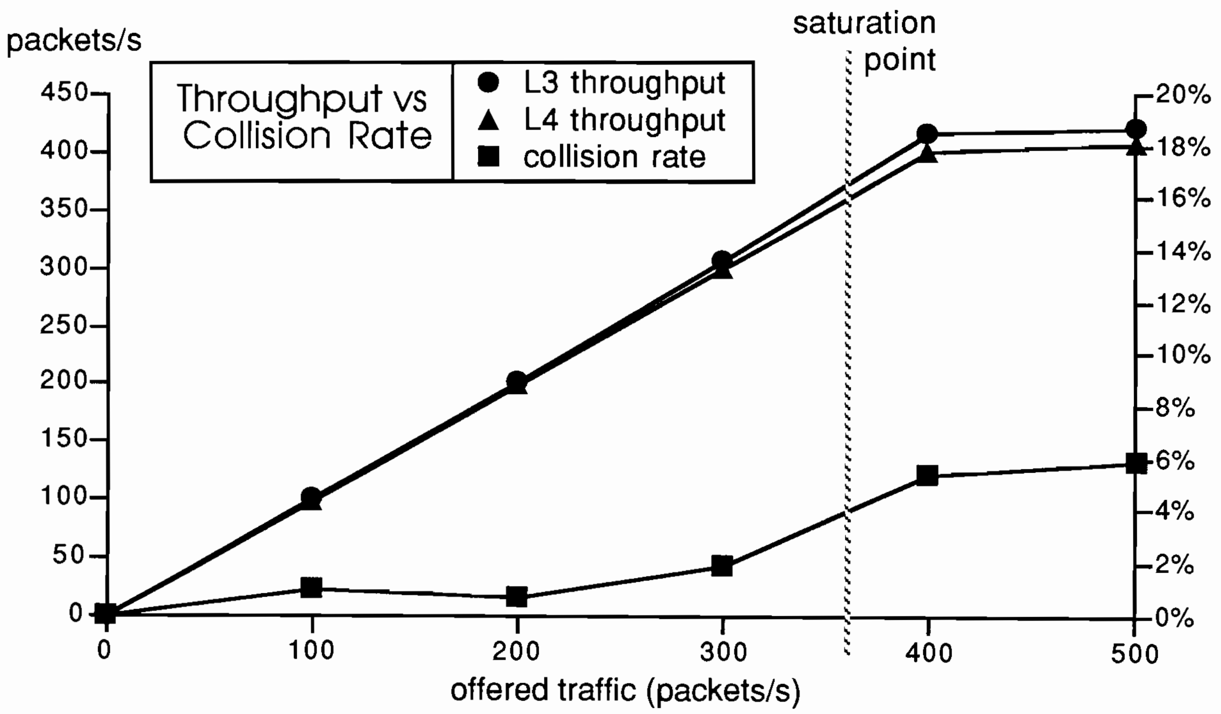

Figure 9E. Acknowledged, Priority, no Collision Detect.

A final comment is that the L4 retry timer would normally be set lower for a network variable connection using priority (this was not done here because of the desire to reduce the variables among the various experimental runs). Thus the occasional message that resulted in a collision would be retried sooner than in these examples.

Experiment F (Effect of Collision Detection on Response Time)

For the collision detection experiment, the baseline configuration (experiment D) was modified so that the two nodes under test made use of collision detection hardware. The characteristics of the traffic generator nodes, and all timers and retry counts remained unmodified. The baseline tests, re-run under these conditions, yielded the charts shown in figures 7F, 8F, and 9F.

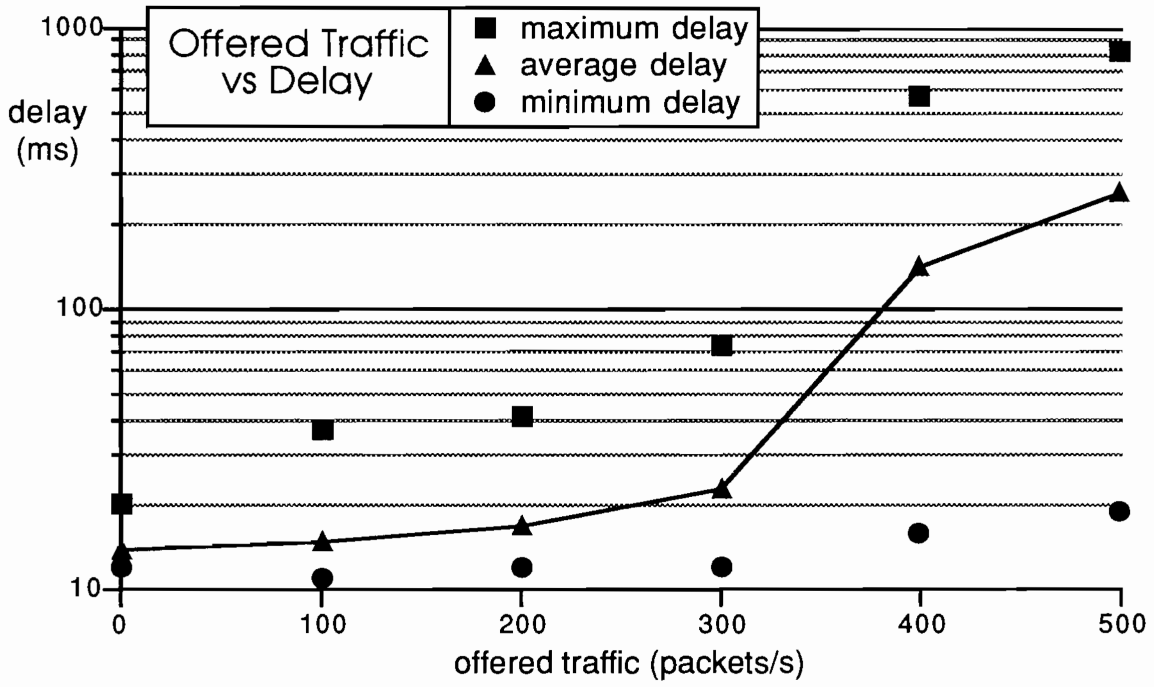

Figure 7F. Acknowledged, no Priority, Collision Detect.

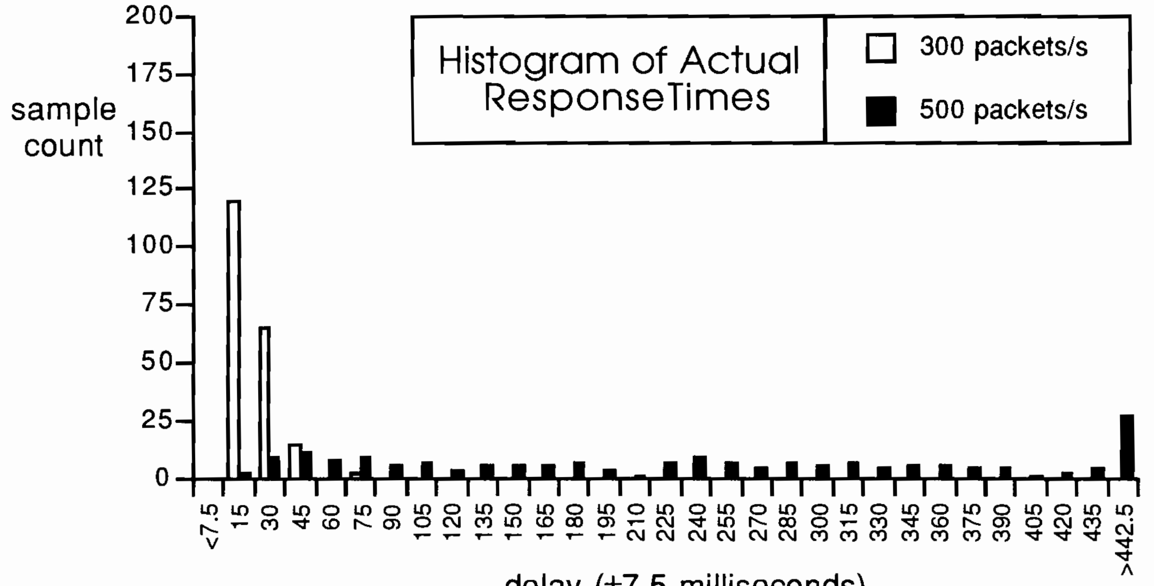

Figure 8F. Acknowledged, no Priority, Collision Detect.

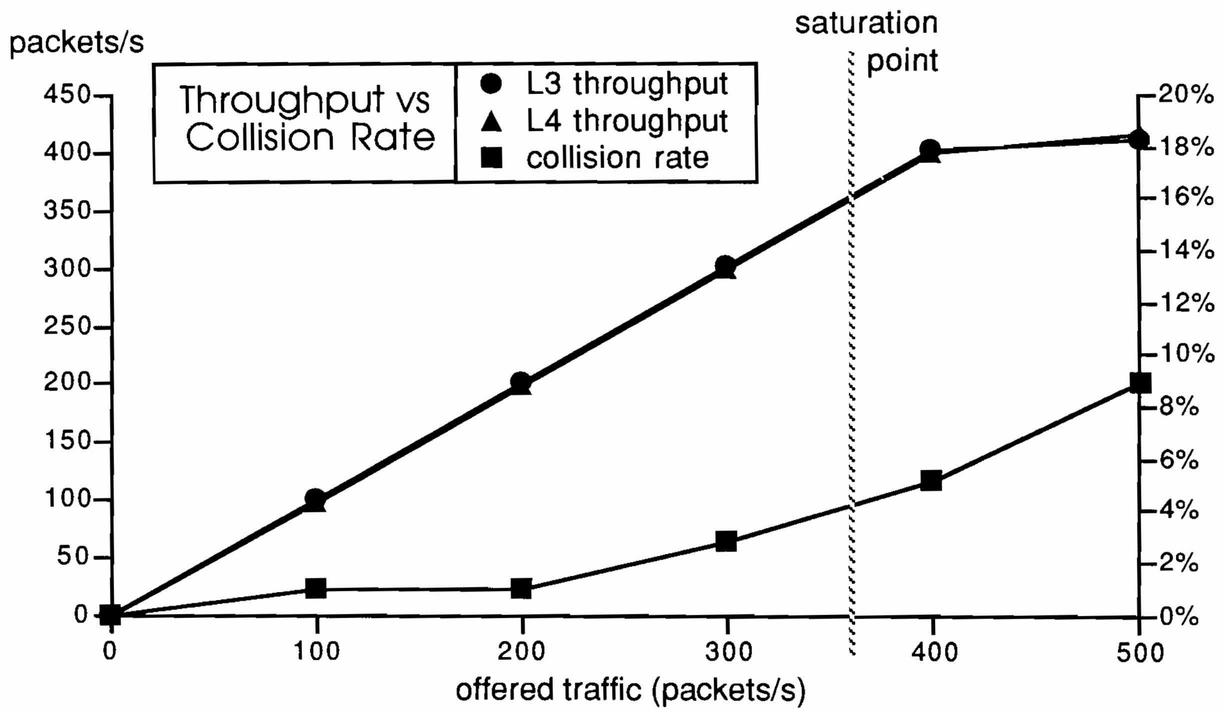

Figure 9F. Acknowledged, no Priority, Collision Detect.

These data clearly show the expected result: a decrease in worse-case response time (cf. figure 7F compared with baseline figure 7D). They also illustrate a generally nonintuitive effect: a comparison of figures 9F and 9D indicates a noticeably increased collision rate above network saturation (although it remains below 4% prior to saturation). This is due to the increased media access delay incurred as a result of increased collisions, a byproduct of collision detection when the network is saturated.

Note that the L3 and L4 throughput rates are equivalent in both this and the following experiments; this is because the protocol is trying the same message again (due to the detected collision), rather than retrying the message later (as would result from a timeout).

Experiment G (Combining Priority and Collision Detection)

This experiment measures the combined effects of priority and collision detection. The baseline configuration (experiment D) was again modified so that the two nodes under test were assigned priority (as in experiment E); in addition, the collision detection hardware used in experiment F was enabled. The characteristics of the traffic generator nodes, and all timers and retry counts remained unmodified. The baseline tests, re-run under these conditions, yielded the charts shown in figures 7G, 8G, and 9G.

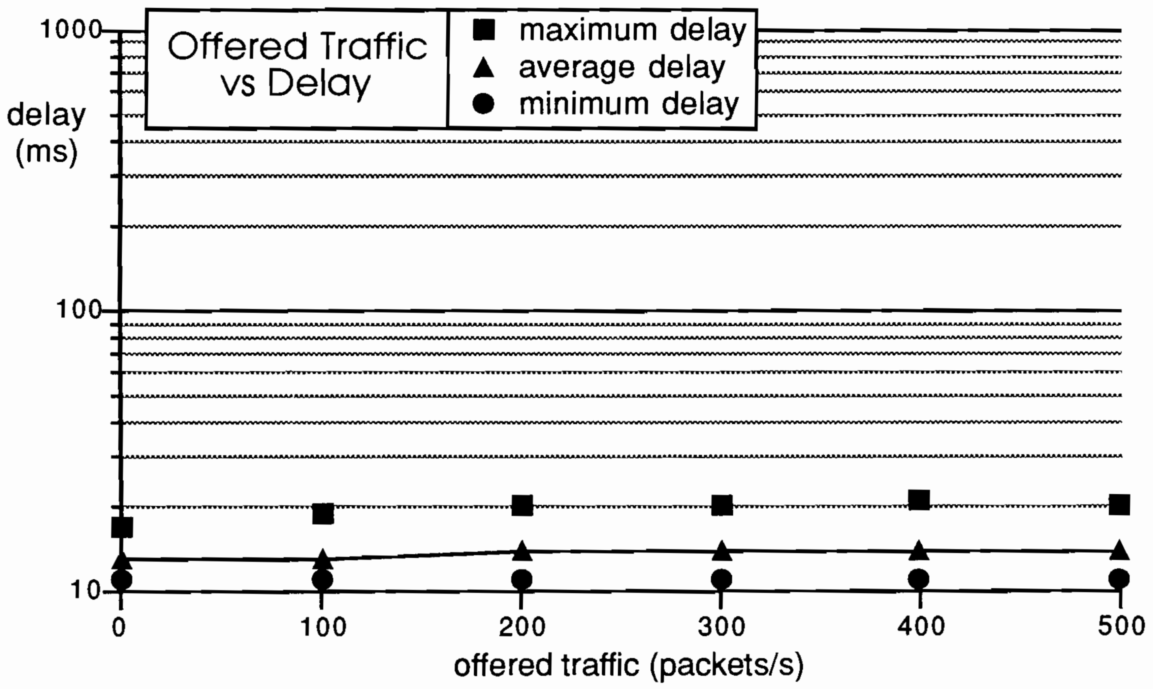

Figure 7G. Acknowledged, Priority, Collision Detect.

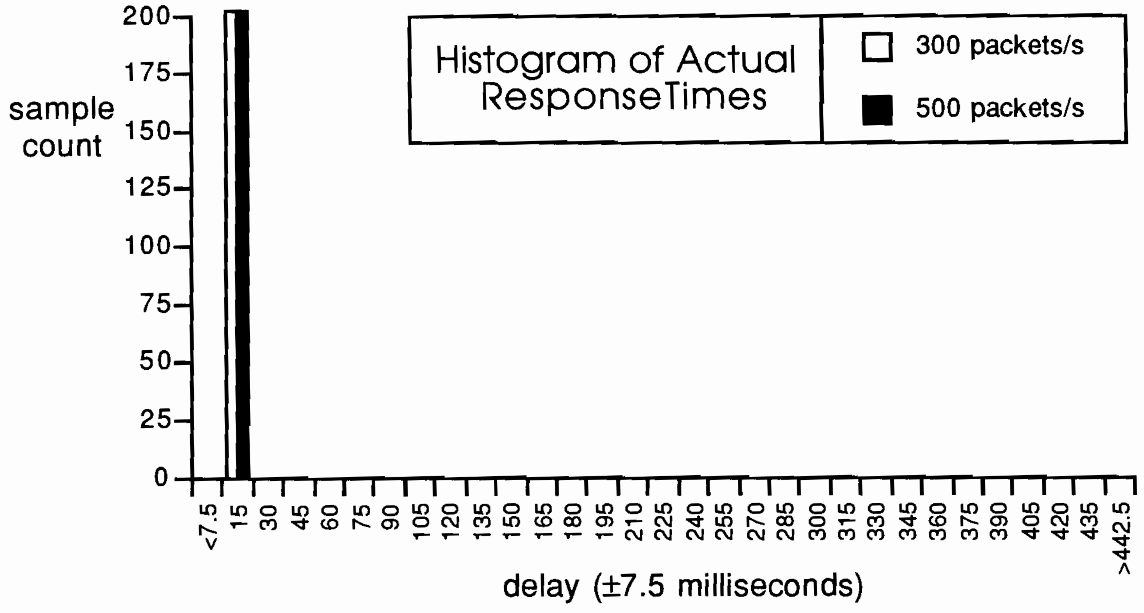

Figure 8G. Acknowledged, Priority, Collision Detect.

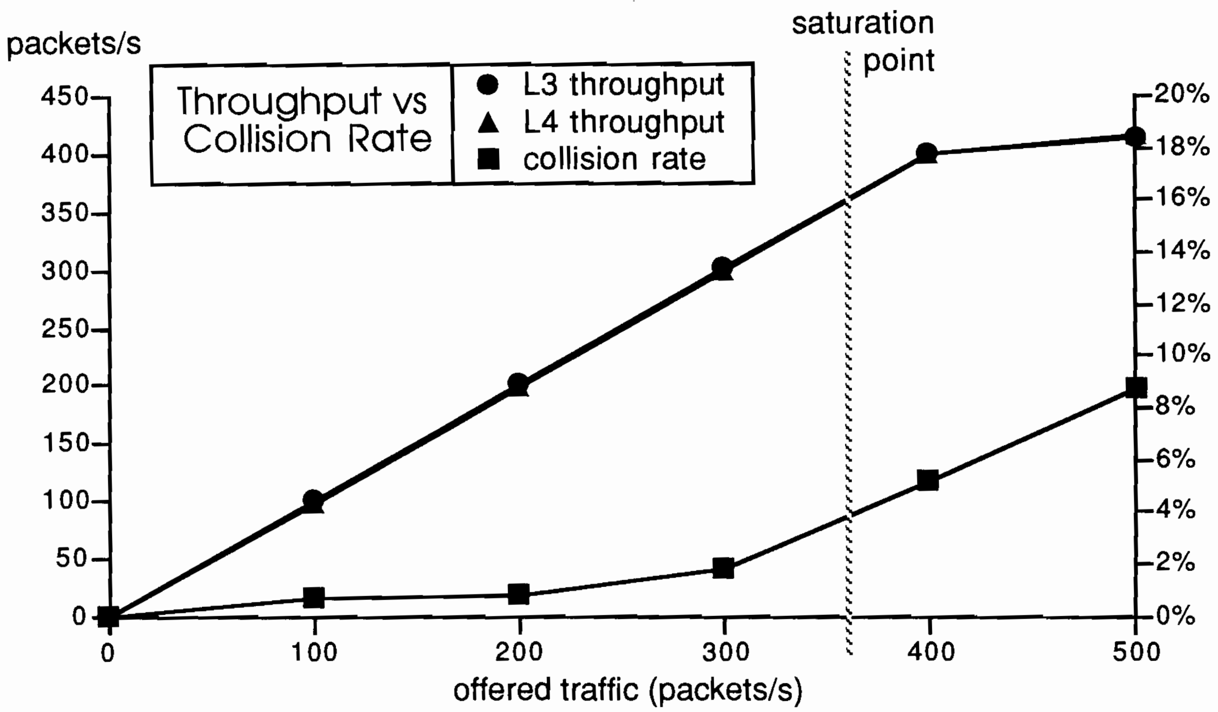

Figure 9G. Acknowledged, Priority, Collision Detect.

As might be anticipated, the combination of priority messages and collision detection results in performance that combines the characteristics of both services.

Figures 7G and 8G are much the same as figures 7E and 8E, with the elimination of the few stray packets delayed by undetected collisions in the latter case. Figure 9G closely imitates figure 9F, though with a slightly lower collision rate due to the added presence of priority services.

Evaluating the Need for Collision Detection Hardware

Many applications can stand an occasional retry as long as the *first* retry gets through. This may be possible with priority alone (i.e., without the use of collision detection), as long as collisions are infrequent. Some collisions occur even when the priority mechanism is in use, as shown by experiment E. For this test, 0.2% (two out of 1000) of the priority packets required an L4 retry.

Collectively, the above data show that using priority dramatically improves the average response time. They also show that the worst-case response time is bounded by the use of priority to at most one retry when the network is below saturation. Finally, they demonstrate the incremental value of collision detection when used in conjunction with priority. This test showed 99.8% of all priority packets getting through on the first try even when the network is well into saturation. Applications for which this percentage of success is too low need the addition of collision detection hardware (which remove the L4 retry delays by detecting collisions and completing retries without waiting for the transaction timers to expire).

Experiments D', E', F', and G' (Effects of Authentication)

These experiments duplicate experiments D–G, substituting authenticated message services for the default (acknowledged) services used by the sender node. The characteristics of the traffic generator nodes and all timers remained unmodified. The retry counts were increased from 4 to 8. The four tests, re-run under these conditions, yielded the following twelve graphs:

7D', 8D', 9D':	basic authenticated service
7E', 8E', 9E':	authenticated, priority service
7F', 8F', 9F':	authenticated service with collision detection
7G', 8G', 9G':	authenticated, priority service with collision detection

Note that the response time histograms for the authenticated message experiments differ slightly from those previously shown; the bin sizes (shown on the X-axis) have been increased from 15 to 50 ms to account for the longer transaction times involved.

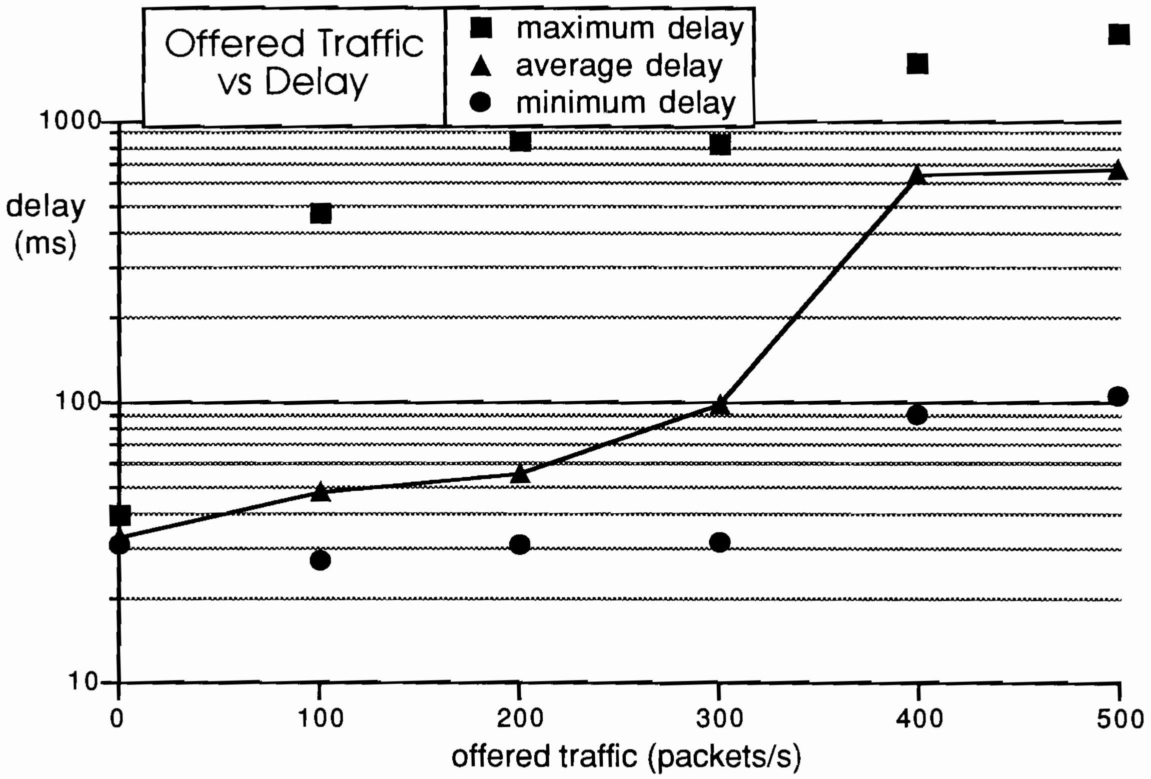

Figure 7D'. Authenticated, no Priority, no Collision Detect.

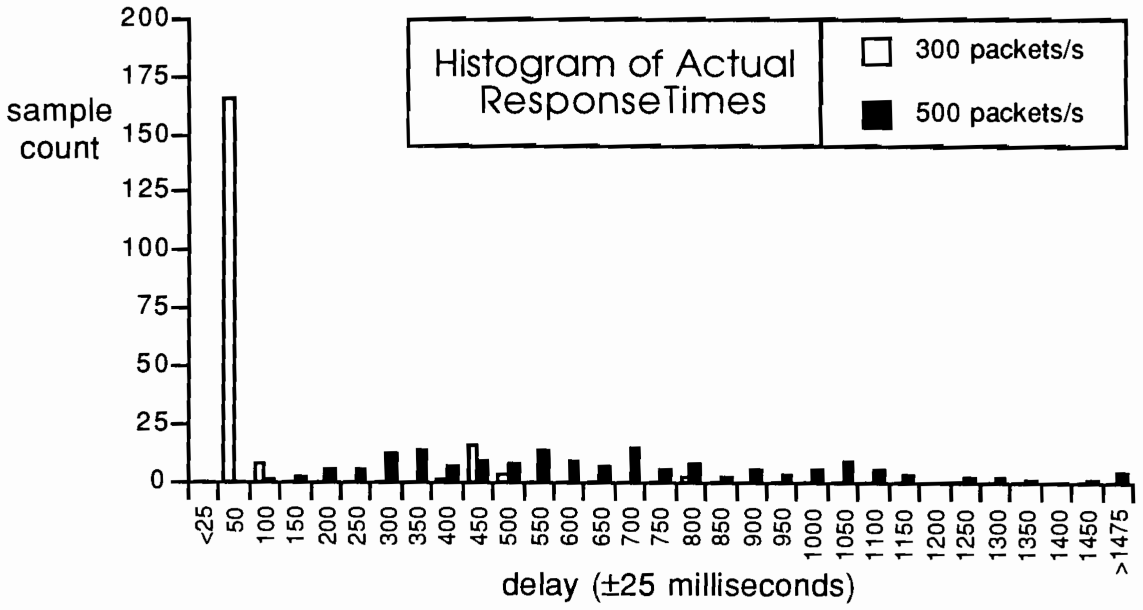

Figure 8D'. Authenticated, no Priority, no Collision Detect.

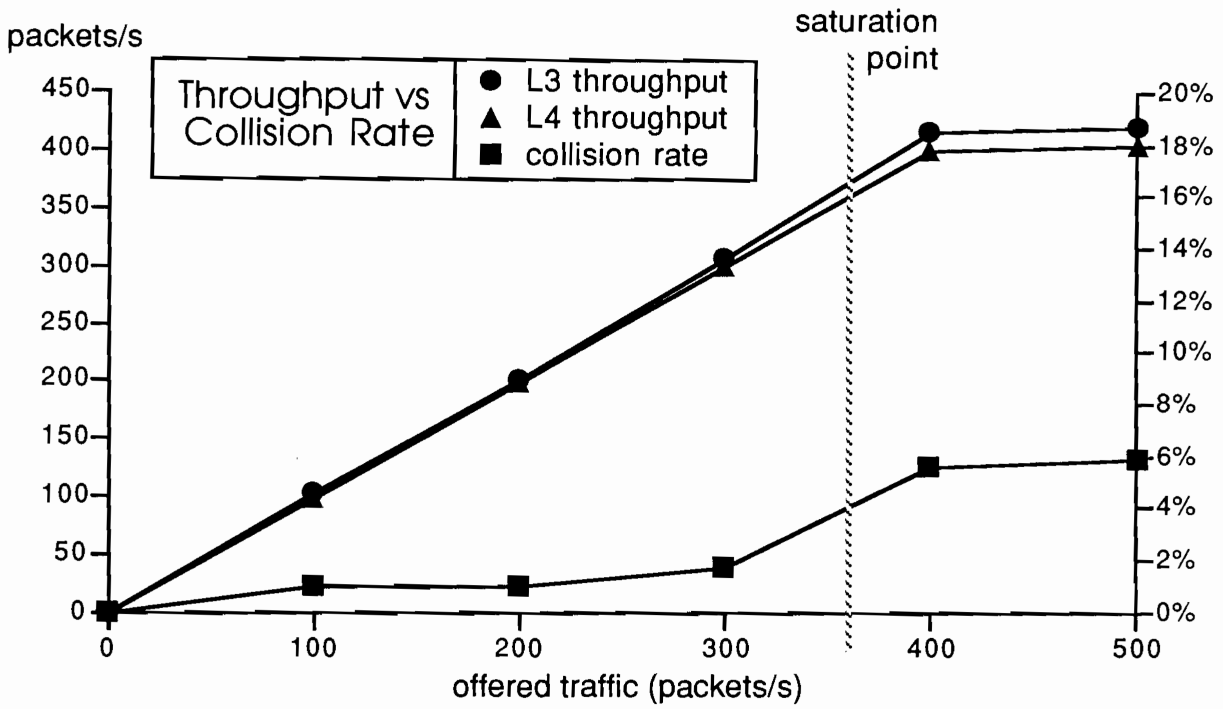

Figure 9D'. Authenticated, no Priority, no Collision Detect.

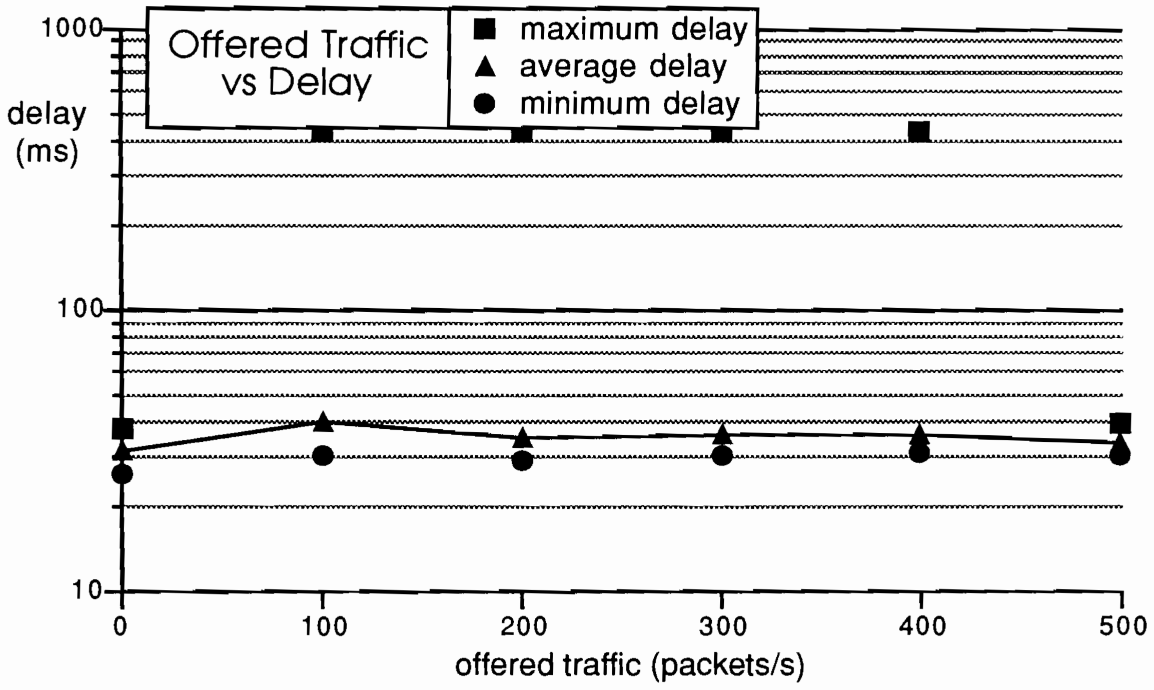

Figure 7E'. Authenticated, Priority, no Collision Detect.

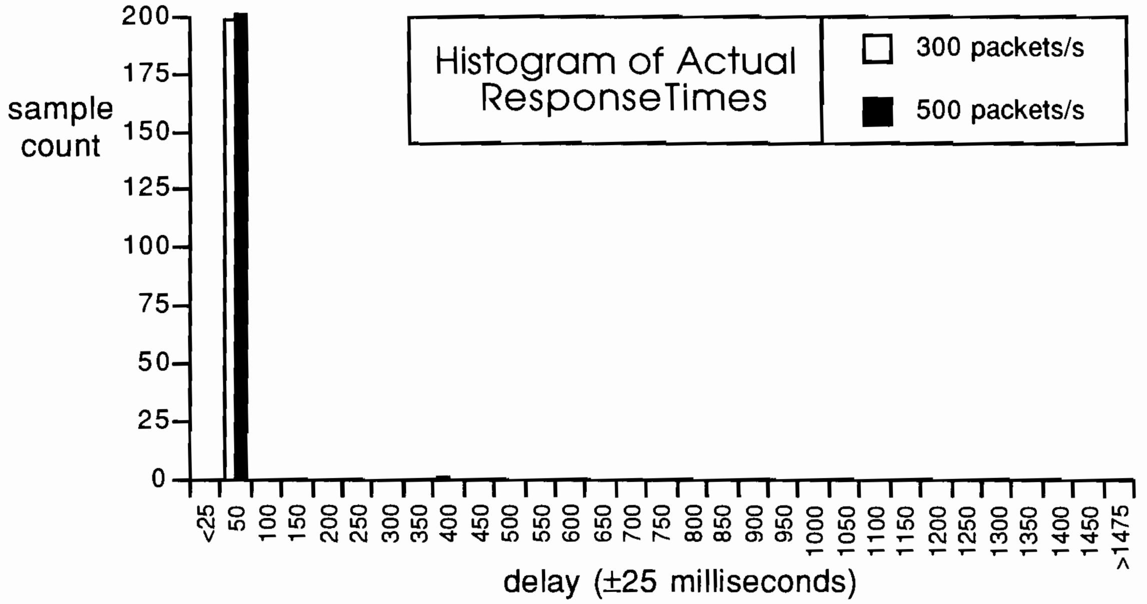

Figure 8E'. Authenticated, Priority, no Collision Detect.

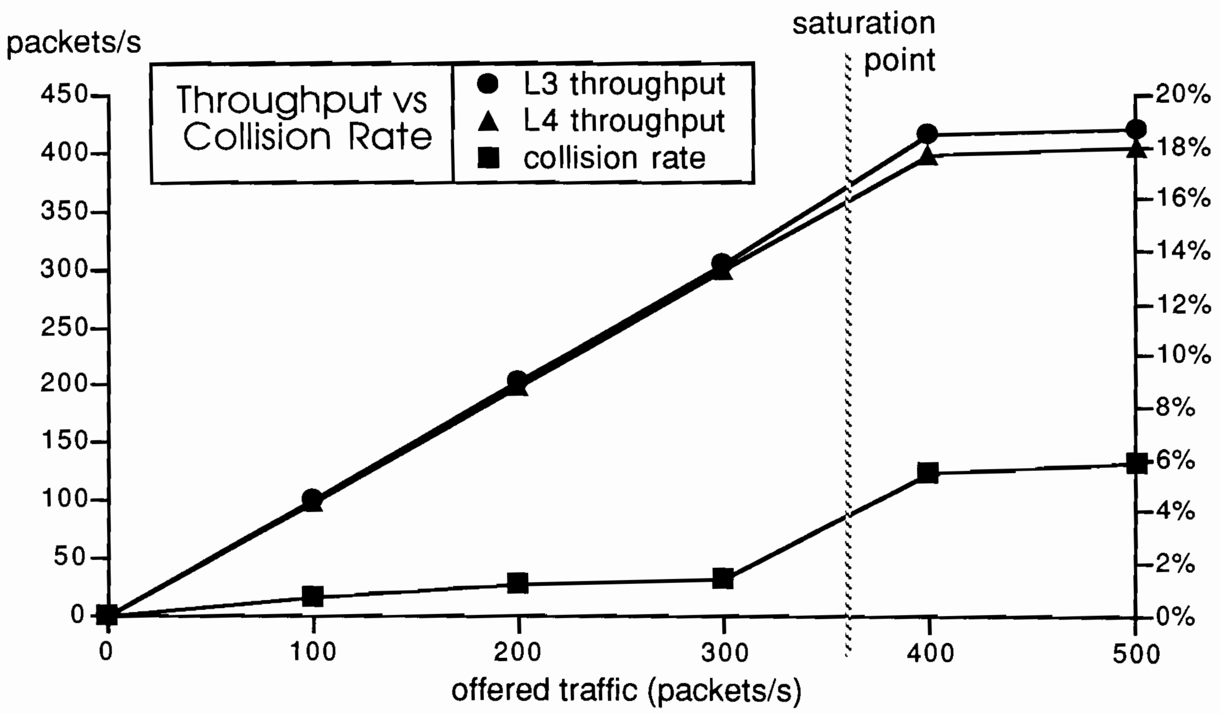

Figure 9E'. Authenticated, Priority, no Collision Detect.

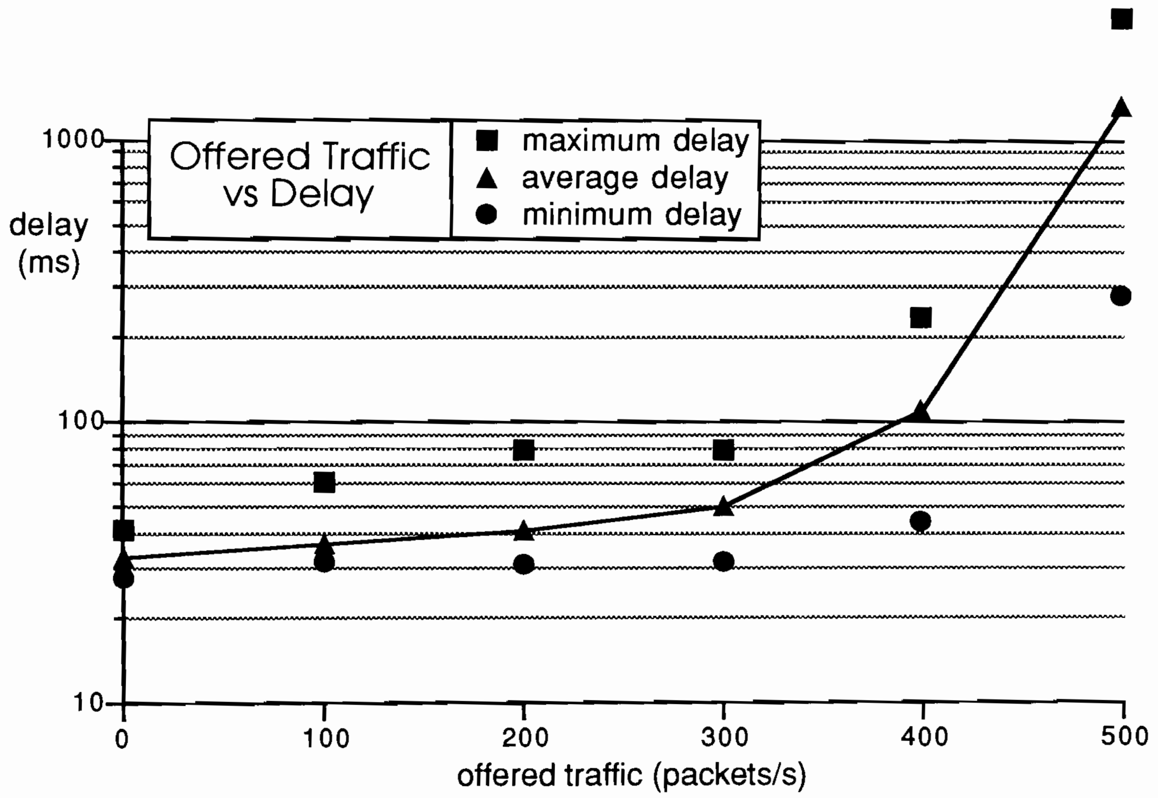

Figure 7F'. Authenticated, no Priority, Collision Detect.

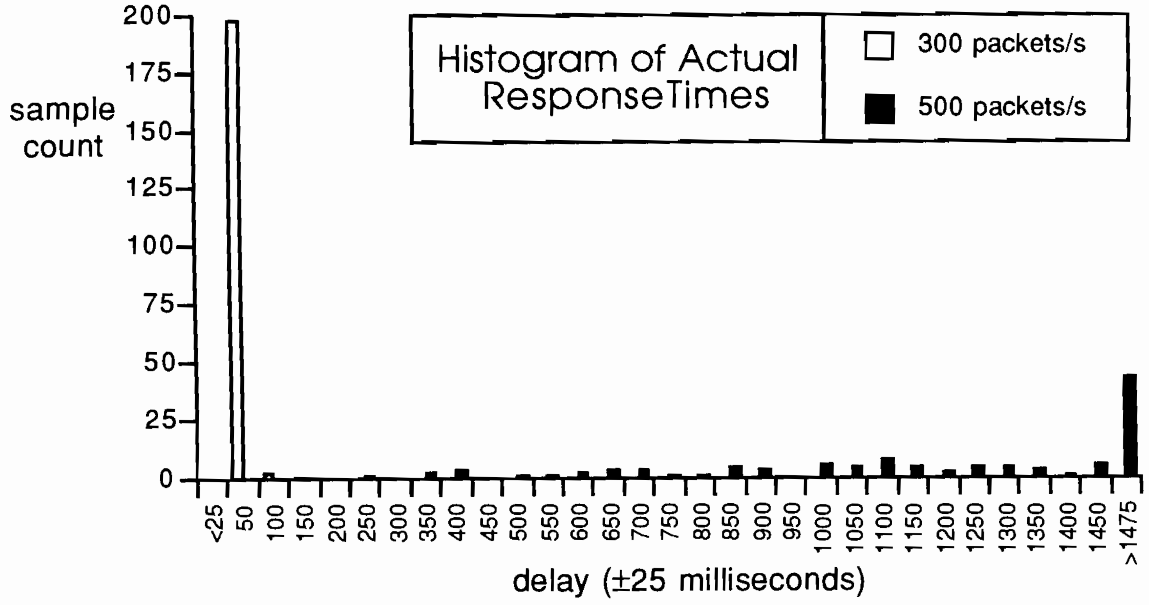

Figure 8F'. Authenticated, no Priority, Collision Detect.

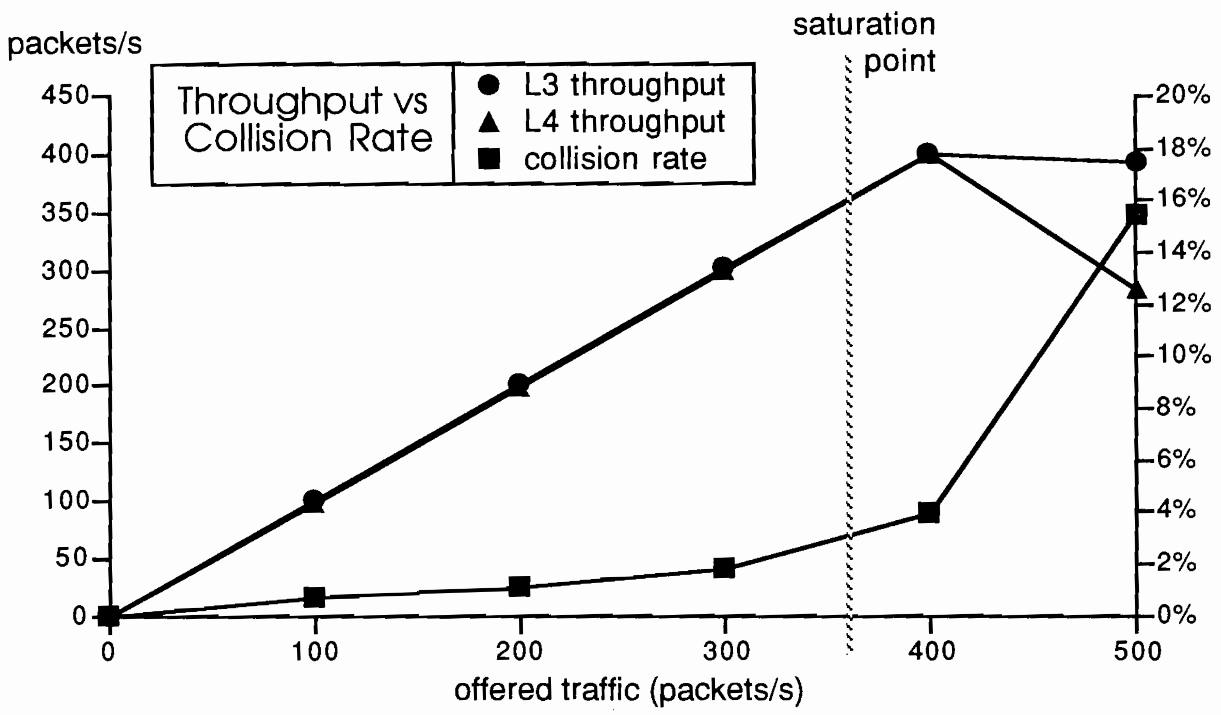

Figure 9F'. Authenticated, no Priority, Collision Detect.

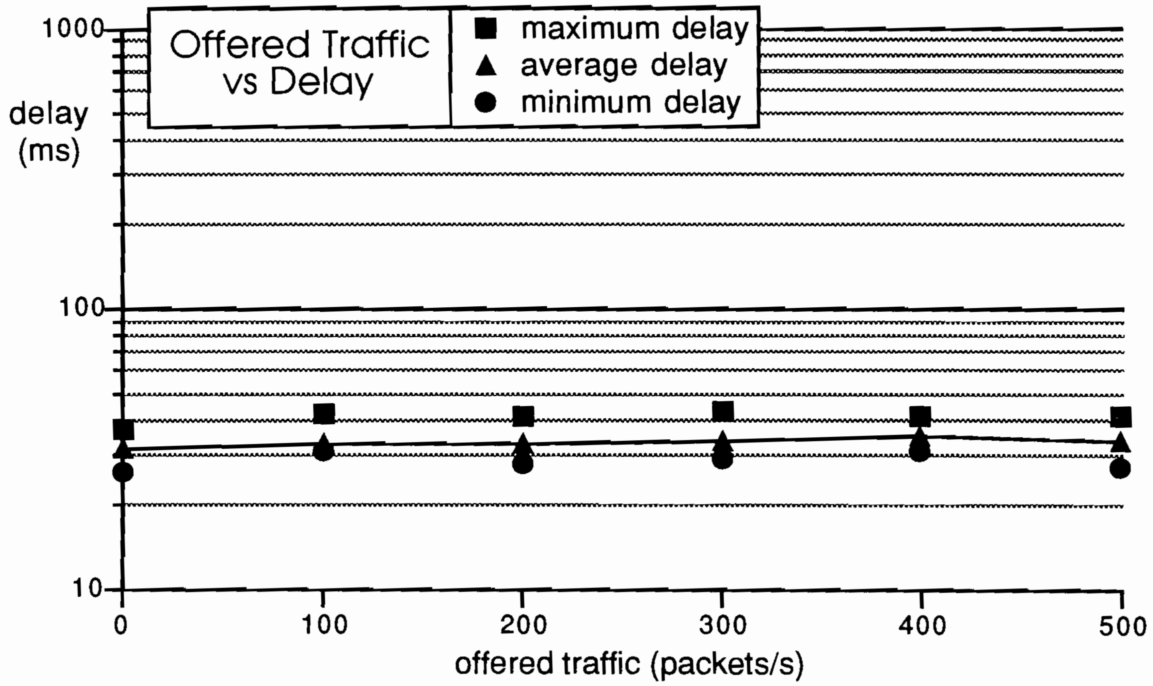

Figure 7G'. Authenticated, Priority, Collision Detect.

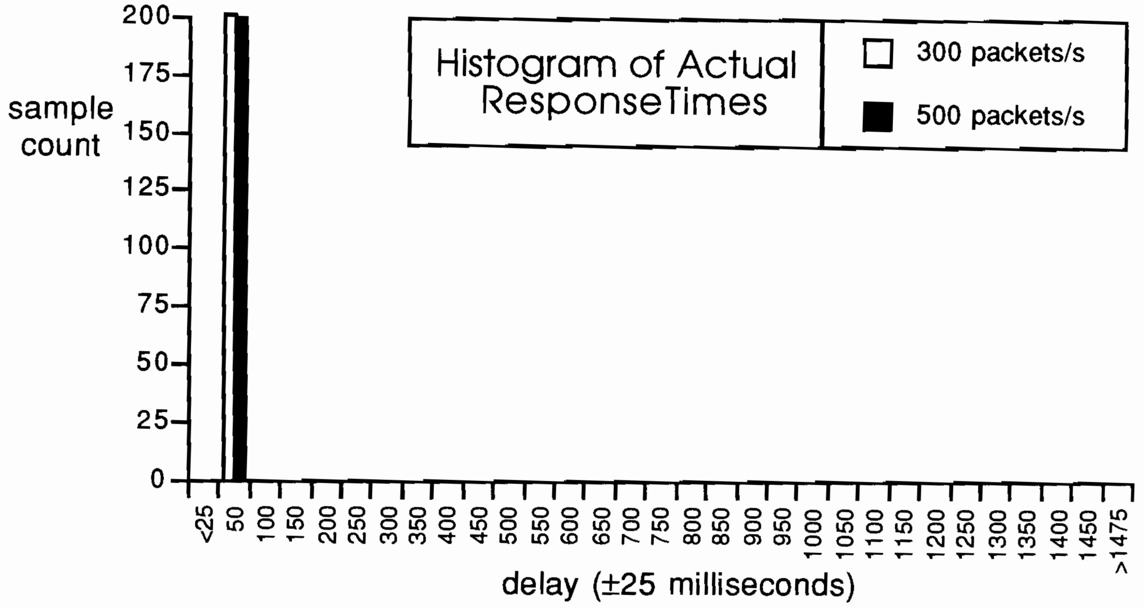

Figure 8G'. Authenticated, Priority, Collision Detect.

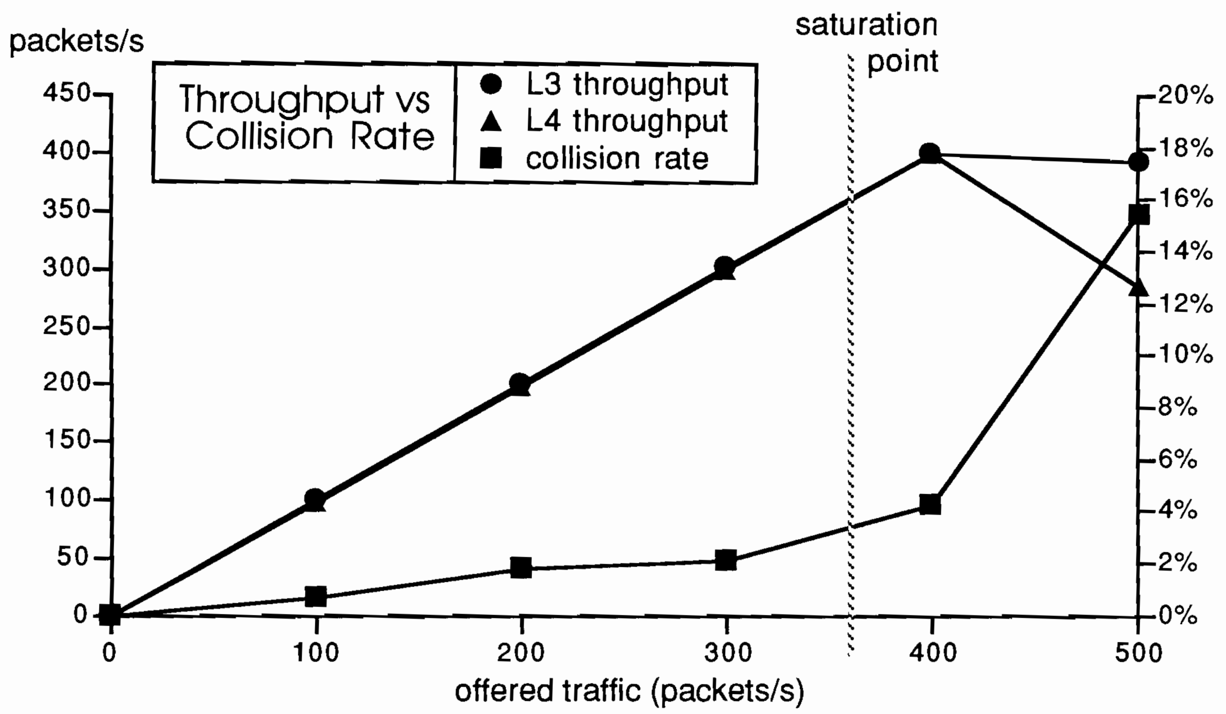

Figure 9G'. Authenticated, Priority, Collision Detect.

The results of the authenticated message experiments closely mirror those for acknowledged messages, allowing for the additional transaction times resulting from the nature of authenticated message services. The only unfamiliar results are shown in figures 9F' and 9G', where the L4 throughput is seen to drop off sharply above the network saturation level. This is due to the sequence of four messages implicit in authenticated transactions, which results in retries caused by intervening timers (unlike the situation shown in figures 9F and 9G, which was discussed earlier).

Experiments H & I (Performance Effects of Collision Avoidance)

As has been noted in other literature about LONTALK's predictive, *p*-persistent MAC sublayer, the ability to avoid collisions is directly related to the predictability of the traffic on the network (see Echelon's engineering bulletins entitled *Enhanced Media Access Control with Echelon's LONTALK Protocol*, and *Optimizing LONTALK Response Time*). When unacknowledged services are used exclusively, the traffic is completely unpredictable. When acknowledged services are used, the traffic is at least 50% predictable (each message produces an expected acknowledgment). When acknowledged services are used in conjunction with multicast addressing, the traffic prediction algorithm works even better (each message produces multiple expected acknowledgments). The quality of prediction further improves with the size of the multicast groups (as the number of expected acknowledgments increases accordingly).

This pair of experiments demonstrates the effects of collision avoidance on performance. By modifying the protocol services listed above, collision avoidance was both defeated (experiment H) and enhanced (experiment I); the resulting data are then compared in figures 10H and 10I.

For experiment H, the test was run with unacknowledged services in the traffic generator nodes (the node under test—the sender—continues to use acknowledged messages to check for failures), thus effectively defeating the collision avoidance algorithm. Figure 10H shows the resulting throughput and collision rates for various levels of offered traffic. This is similar to the figure 9 graph, though the axis values have been considerably increased (as we are primarily interested in the performance characteristics above network saturation). Note in particular the collision rate climbing to over 50%, due to unpredictable traffic produced by the unacknowledged messages. L4 throughput (which is identical to L3 throughput in the case of unacknowledged messages) is also seen to drop off dramatically under conditions of high load. And finally, it should be noted that this situation began to produce message failures, beginning at an offered traffic rate of 800 packets/s (this even after 16 transmission attempts, the most permitted by the protocol). Although even in this extreme situation the rate of such failures was low (below 0.5%), it nonetheless indicates a mode of operation that cannot be tolerated by many applications.

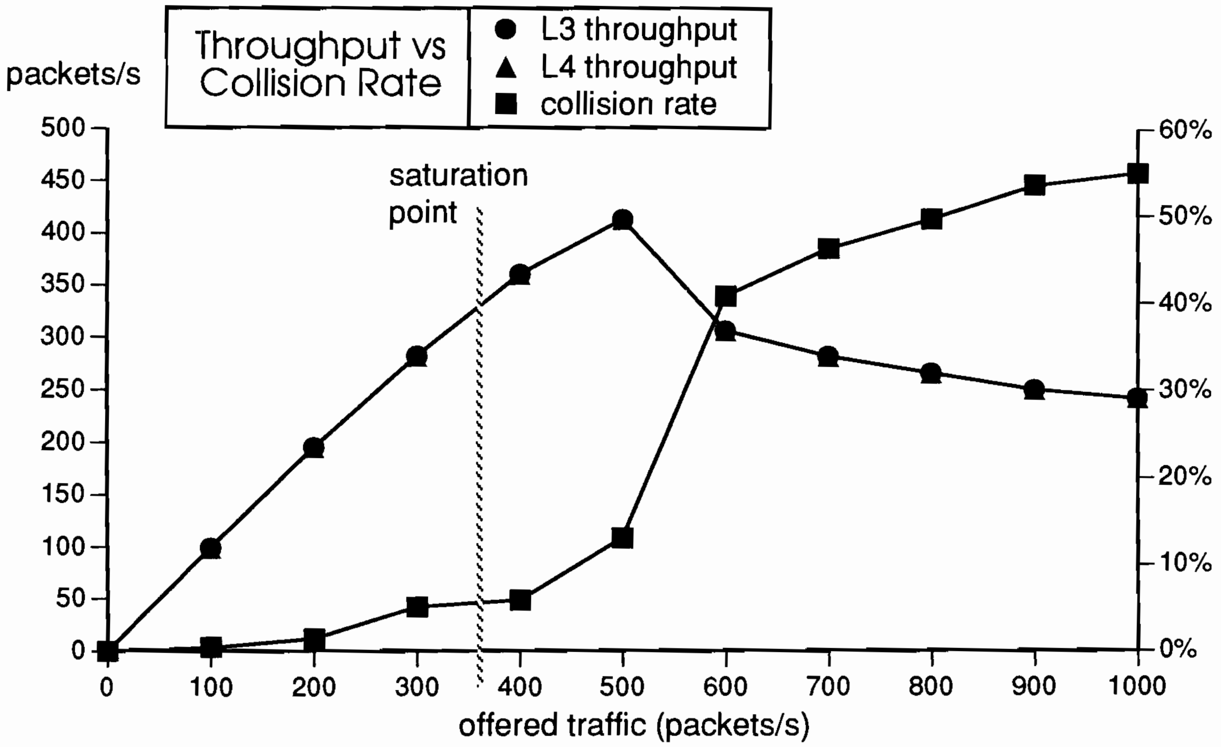

Figure 10H. Unacknowledged, no Priority, no Collision Detect.

Experiment I is a duplication of H, substituting acknowledged services for all traffic generator activity, and organizing the traffic generator nodes into groups of two (unicast message transactions). Even these minimal opportunities for collision avoidance, however, produced an informative contrast with the results of experiment H. Figure 10I shows the throughput and collision rates for this final experiment.

Note that now the collision rate is seen to stabilize at less than a fifth (10% versus 55%) of the rate seen in figure 10H, even as the offered traffic increases well beyond the saturation level. Throughput also stabilizes, with an L4 rate of about 250 packets/s (a slight drop from its pre-saturation value of 360 packets/s); in this case this is less than L3 throughput (which stabilizes at approximately 380 packets/s), due to the retries made necessary by the many collisions. Also (and unlike the situation in experiment H) there were no message failures. The worst case response time, observed at an offered traffic level of 800 packets/s, was 2371 ms (though even in this heavily overloaded situation, an average response time of 389 ms was maintained).

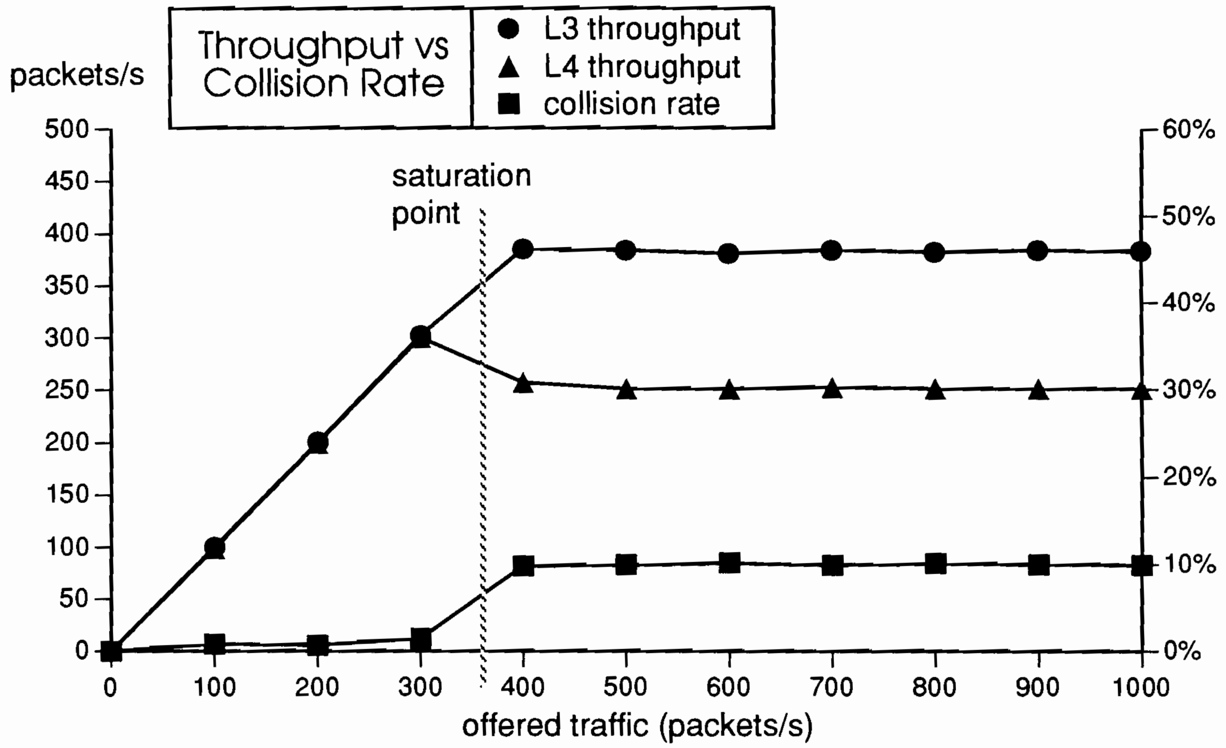

Figure 10I. Acknowledged Unicast, no Priority, no Collision Detect.

Summary

A significant amount of empirical network throughput data under various conditions of message service and media access control is summarized in the foregoing sections. Key conclusions from these experiments are stated below:

- The maximum measured transaction rate of a single node is 326 transactions per second. It occurs using unacknowledged network variable message services with the network operating at a bit transfer rate of 1.25 Mbits/s, and a data block size of one byte. The equivalent acknowledged rate is 69 transactions/s; this rate is much lower because time is lost waiting for message acknowledgments.
- The maximum measured data transfer rate of a single node is 203 kbits/s. It occurs using unacknowledged explicit message services, with a data block size of 228 bytes, and assumes that the copying of the message is done by an external processor. The equivalent acknowledged rate is 73 kbits/s. If the message copying is performed by the NEURON CHIP itself (i.e., by the application processor), the rates are 35 and 27 kbits/s respectively, due to the extra overhead imposed by copying the large data block. The network is assumed to be operating at a bit transfer rate of 1.25 Mbits/s in all cases.

- The best-case response time in a LONTALK network is 5.5 milliseconds, achieved using unacknowledged explicit messages at 1.25 Mbits/s, and assumes that the copying of the message is done by an external processor. The equivalent acknowledged time is 12 ms.
- Priority is an effective way to reduce average response time, although it does not completely eliminate the possibility of a collision.
- Collision detection hardware can effectively constrain worst case response times, though it generally offers no other performance improvements to networks operating below saturation. Above saturation, collision detection often increases the collision rate.
- The predictive collision avoidance scheme used in the LONTALK protocol provides superior sustained throughput. When message traffic volumes are even minimally predictable, the number of collisions and retries drops by a factor of five in overloaded networks.

The methodology presented in this engineering bulletin can be used to measure response time and throughput on customer-designed LONWORKS networks.

ECHELON

Part Number 005-0010-01
Rev. A

Echelon Corporation
4015 Miranda Avenue
Palo Alto, CA 94304
Telephone (415) 855-7400
Fax (415) 856-6153

Echelon Europe Ltd
105 Heath Street
London NW3 6SS
England
Telephone (071) 431-1600
Fax (071) 794-0532
International Telephone + 44 71 431-1600
International Fax + 44 71 794-0532

Echelon Japan K.K.
AIOS Gotanda Building #808
10-7, Higashi-Gotanda 1-chome,
Shinagawa-ku, Tokyo 141, Japan
Telephone (03) 3440-8638
Fax (03) 3440-8639